**TUGAS AKHIR – TE141599**

# PENGEMBANGAN METODE PENGAMBILAN OBJEK BERBAHAYA SECARA OTOMATIS PADA *MOBILE ROBOT*

Mohammad Nasrul Mubin
NRP 2212 100 113

Dosen Pembimbing
Ronny Mardiyanto, ST., MT., Ph.D
Dr. Ir. Djoko Purwanto, M.Eng.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

**FINAL PROJECT – TE141599**

# DANGEROUS OBJECT AUTOMATICALLY GRASPING METHOD DEVELOPMENT ON MOBILE ROBOT

Mohammad Nasrul Mubin
NRP 2212 100 113

Advisor
Ronny Mardiyanto, ST., MT., Ph.D
Dr. Ir. Djoko Purwanto, M.Eng.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Industry Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

# PENGEMBANGAN METODE PENGAMBILAN OBJEK BERBAHAYA SECARA OTOMATIS PADA *MOBILE ROBOT*

## TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui :

| Dosen Pembimbing I, | Dosen Pembimbing II, |
|---|---|
| Ronny Mardiyanto, S.T., M.T.,Ph.D. | Dr. Ir. Djoko Purwanto, M.Eng. |
| NIP. 198101182003121003 | NIP. 196512111990021002 |

SURABAYA
JUNI, 2016

**PENGEMBANGAN METODE PENGAMBILAN OBJEK BERBAHAYA SECARA OTOMATIS PADA *MOBILE ROBOT***

**Mohammad Nasrul Mubin**
**2212100113**

**Dosen Pembimbing I : Ronny Mardiyanto, ST., MT., Ph.D**
**Dosen Pembimbing II : Dr. Ir. Djoko Purwanto, M.Eng.**

# ABSTRAK

Penjinakan bom yang dilakukan manusia secara kontak langsung memiliki risiko yang sangat besar. Risiko tersebut seperti meledaknya objek tersebut hingga melukai bahkan membunuh sang penjinak bom. Karena hal inilah robot pengambil objek perlu direalisasikan.

Pada penelitian ini, direalisasikan suatu *mobile robot* autonomous pengambil objek berbahaya sebagai pengembangan penelitian sebelumnya. Fitur robot pada penelitian kali ini telah dikembangkan hingga mampu mengambil objek dalam pose apapun. Dengan memanfaatkan tangkapan gambar dari kamera dan terbatas dengan objek berwarna merah, robot mampu mengestimasi orientasi objek. Estimasi dimulai dengan mengkonversi gambar RGB ke lingkup HSV. Dari gambar HSV, diaplikasikan filter warna merah dan morfologi. Aplikasi deteksi tepi digunakan untuk mempermudah pengaplikasian *hough transform.* Nilai x dan y dari fungsi tersebut digunakan untuk menghitung nilai orientasi objek. Nilai ini kemudian diproses untuk diterapkan pada pergerakan *gripper*.

Beberapa hasil pengujian pada tugas akhir kali ini antara lain masih terdapat kesalahan pada deteksi orientasi yaitu mencapai 7,7%, terdapat batas bawah nilai iluminansi agar sistem dapat bekerja yaitu 7,1, pergerakan *gripper* sesuai dengan kalkulasi dari orientasi yang terdeteksi, dan lama pengambilan dipengaruhi oleh jarak serta orientasi objek.

Kata kunci : mobile robot, filter warna, estimasi orientasi.

*Halaman Ini Sengaja Dikosongkan*

***DANGEROUS OBJECT AUTOMATICALLY GRASPING METHOD DEVELOPMENT ON MOBILE ROBOT***

**Mohammad Nasrul Mubin**
**2212100113**
**Advisor 1 : Ronny Mardiyanto, ST., MT., Ph.D**
**Advisor 2 : Dr. Ir. Djoko Purwanto, M.Eng.**

# *ABSTRACT*

Manually defusing a bomb posseses a high risk. It is such as when the bomb explode, the operator can get injured or even killed. Because of this object grasping robot needs to be realized.

In this research, it is realized an autonomous dangerous objects grasping mobile robot. Robot features in the present research has been developed to be able to take the object in any pose. By utilizing capture images from the camera and limited to a red object, the robot is able to estimate the pose of the object. Estimates begins with converting RGB image to HSV image. From HSV image, the red color filter and morphology function is applied. Application of edge detection is used to facilitate the application of hough transform function. X and y values of the function used to calculate the value of object orientation. This value is then processed to apply to the movement of gripper.

Some of the test results inter alia there are errors in the orientation detection, reaching 7.7%, there is a lower limit value of illuminance for the system to work is 7.1, there is no error in the movement of the gripper based on the orientation's calculation, and the time of the grasping object is influenced by the distance and the object poses.

Keyword : mobile robot, color filter, pose estimation.

*Halaman Ini Sengaja Dikosongkan*

# KATA PENGANTAR

Segala puji bagi Allah yang merajai alam semesta atas segala nikmat, berkah, dan hidayah-Nya yang tak terhitung kepada penulis, hingga penulis mampu menyelesaikan Tugas Akhir ini. Penulis ingin mengabadikan ucapan terimakasih kepada seluruh pihak yang telah membantu penyempurnaan tugas akhir didalam kata pengantar yang singkat ini. Semoga kasih dan sayang dari Allah selalu tercurahkan kepada:

1. Nabi Muhammad SAW, yang selalu menjadi sosok panutan penulis, termasuk dalam menghadapi tantangan hidup.
2. Keluarga penulis yang tiada henti mendukung penulis.
3. Bapak Ronny Mardiyanto, ST., MT.,Ph.D. dan Dr. Ir. Djoko Purwanto, M.Eng. yang tidak pernah lelah dan selalu sabar membimbing penulis.
4. Seluruh dosen bidang studi elektronika.
5. Seluruh keluarga Arek Pekalongan yang terus-menerus memberi dukungan mental dan fisik.
6. Mas, mbak, dan rekan seperjuangan BEM FTI ITS 2014/2015 maupun BEM FTI ITS 2013/2014 yang selalu menjadi keluarga yang berharga dan membuat bahagia selama di perantauan.
7. Rekan Eks Kontingen Kota Pekalongan dalam Jambore Daerah Jawa Tengah yang silaturahimnya tetap terjaga sampai sekarang dan tetap bergerak di sektor positif.
8. Rekan E52 sebagai teman seperjuangan dalam segala tantangan didunia pergulatan yang sekarang mulai gandrung disebut dengan masa mahasiswa.

Penulis menyadari bahwa tugas akhir ini penuh dengan kekurangan. Penulis sangat menghargai kritik dan saran untuk menutup kekurangan yang tercipta untuk disempurnakan.

Surabaya, Juni 2015

Penulis

*Halaman Ini Sengaja Dikosongkan*

# DAFTAR ISI

**ABSTRAK** ..... **i**
**ABSTRACT** ..... **iii**
**KATA PENGANTAR** ..... **v**
**DAFTAR ISI** ..... **vii**
**DAFTAR GAMBAR** ..... **ix**
**DAFTAR TABEL** ..... **xi**
**BAB I PENDAHULUAN** ..... Error! Bookmark not defined.
1.1 Latar Belakang ..... **Error! Bookmark not defined.**
1.2 Perumusan Masalahan ..... **Error! Bookmark not defined.**
1.3 Tujuan ..... **Error! Bookmark not defined.**
1.4 Batasan Masalah ..... **Error! Bookmark not defined.**
1.5 Metodologi Penelitian ..... 2
1.6 Sistematika Penulisan ..... 3
1.7 Relevansi ..... **Error! Bookmark not defined.**
**BAB II DASAR TEORI DAN TINJAUAN PUSTAKA** ..... Error! Bookmark not defined.
2.1 Dasar Teori ..... **Error! Bookmark not defined.**
2.1.1 OpenCV ..... **Error! Bookmark not defined.**
2.1.2 *Color Detection and Object Tracking* **Error! Bookmark not defined.**
2.1.3 HSV *(Hue, Saturation, Value)* **Error! Bookmark not defined.**
2.1.4 Morfologi ..... 8
2.1.5 Edge Detection ..... 9
2.1.6 Hough Transform ..... **Error! Bookmark not defined.**
2.1.7 Pulse Width Modulation ..... **Error! Bookmark not defined.**
2.1.8 Motor Sevo ..... 13
2.1.9 Komunikasi Serial ..... 15
2.1.10 Sensor Ultrasonik ..... **Error! Bookmark not defined.**
2.1.11 Robot Kinematics ..... **Error! Bookmark not defined.**
2.1.12 USB to TTL ..... 18
2.2 Tinjauan Pustaka ..... **Error! Bookmark not defined.**
2.2.1 Pustaka Pendukung ..... 18

2.2.2 Sistem Awal ..... 26
2.2.2.1 Arduino Uno ..... 28
2.2.2.2 Raspberry Pi 2 ..... 30
2.2.2.3 Arduino Mega ..... 33
**BAB III PERANCANGAN SISTEM ..... 35**
3.1 Diagram Blok Sistem ..... 35
3.2 Pengembangan Sistem ..... 36
3.2.1 Antarmuka ..... 40
3.2.2 Arduino Uno ..... 43
3.2.3 Raspberry Pi 2 ..... 45
3.2.4 Arduino Mega ..... 46
3.2.5 Raspberry Pi 3 ..... 52
3.4 Proses Estimasi Orientasi Objek ..... 58
**BAB IV PENGUJIAN DAN ANALISIS ..... 59**
4.1 Pengujian estimasi orientasi objek ..... 60
4.2 Pengujian sistem menurut intensitas cahaya ..... 62
4.3 Pengujian Pergerakan Griper ..... 63
4.4 Pengujian Lama Pengambilan Objek ..... 64
4.5 Pengujian Jarak Kamera dengan Objek ..... 65
**BAB V PENUTUP ..... 67**
5.1 Kesimpulan ..... 67
5.2 Saran ..... 67
**DAFTAR PUSTAKA ..... 69**
**LAMPIRAN ..... 73**
**BIODATA PENULIS ..... 99**

# DAFTAR GAMBAR

**Gambar 2.1** Warna HSV ..... 6
**Gambar 2.2** Deteksi tepi Canny ..... 10
**Gambar 2.3** Gambar Input dan Hasil Pemetaan ..... 11
**Gambar 2.4** Sinyal PWM ..... 12
**Gambar 2.5** Motor Servo ..... 13
**Gambar 2.6** Pulsa dan pergerakan motor servo ..... 14
**Gambar 2.7** Lengan 4 Dof ..... 17
**Gambar 2.8** USB to TTL ..... 18
**Gambar 2.9** a) bentuk referensi dan model yang diteliti ..... 23
**Gambar 2.9** b) bentuk referensi dan model yang diteliti dengan HT 23
**Gambar 2.9** c)bentuk referensi dan model yang diteliti dengan LS .. 23
**Gambar 2.9** Diagram Sistem Awal ..... 22
**Gambar 2.10** a-c) bentuk referensi dan bentuk deformasi dengan σ 24
**Gambar 2.10** d-f) bentuk referensi dan bentuk deformasi dengan σ dengan HT ..... 24
**Gambar 2.10**e-i) bentuk referensi dan bentuk deformasi dengan σ dengan LS. ..... 24
**Gambar 2.11** a-c) bentuk referensi dan bentuk yang diteliti dengan *outlier* ..... 25
**Gambar 2.11**d-f) bentuk referensi dan bentuk yang diteliti dengan *outlier* dengan HT ..... 25
**Gambar 2.11**g-i) bentuk referensi dan bentuk yang diteliti dengan *outlier* dengan LS ..... 25
**Gambar 2.12** a)Hasil Monte Carlo dengan bentuk deformasi ..... 25
**Gambar 2.12** b)Hasil Monte Carlo dengan *outlier* ..... *25*
**Gambar 2.13** Diagram Alir Sistem Awal ..... 26
**Gambar 2.14** Diagram Alir Arduino Uno ..... 29
**Gambar 2.15** Pengkabelan ESC dengan Arduino Uno dan Motor .... 30
**Gambar 2.16** Pengkabelan Arduino Uno dengan Arduino Mega ..... 30
**Gambar 2.17** Diagram alir Raspberry Pi 2 ..... 31
**Gambar 2.18** Pengkabelan Arduino Uno dengan Raspberry Pi ..... 32
**Gambar 2.19** Gambar kursor (lingkaran merah) ..... 32
**Gambar 2.20** Diagram alir program sistem awal Arduino Mega ..... 34
**Gambar 3.1** Diagram Sistem Robot ..... 35
**Gambar 3.2** Diagram Alir keseluruhan program ..... 38
**Gambar 3.3** Ilustrasi Robot Tampak Samping ..... 39

**Gambar 3.4a** Ilustrasi Robot Tampak Layer 1 ..... 39
**Gambar 3.4b** Ilustrasi Robot Tampak Layer 2 ..... 39
**Gambar 3.4c** Ilustrasi Robot Tampak Layer 3 ..... 39
**Gambar 3.5** Radio Kontrol sebagai antarmuka ..... 41
**Gambar 3.6** Radio receiver, Video sender dan Video receiver ..... 42
**Gambar 3.7** Hasil LCD Monitor ..... 42
**Gambar 3.8** Mode Robot Pada Radio Kontrol ..... 42
**Gambar 3.9** Ilustrasi gerakan kamera (kanan) diiringi dengan input dari remot (kiri) ..... 43
**Gambar 3.10** Arduino Uno ..... 44
**Gambar 3.11** Arduino Mega ..... 46
**Gambar 3.12** Diagram alir Arduino Mega ..... 47
**Gambar 3.13** Pengkabelan Arduino Mega dengan receiver, multiplexer dan sensor ultrasonik ..... 48
**Gambar 3.14** Pengkabelan Arduino Mega dengan motor servo pada kamera analog dan lengan robot ..... 48
**Gambar 3.15** Peletakan enam motor servo dan kamera ..... 49
**Gambar 3.16** Rincian ukuran lengan robot ..... 49
**Gambar 3.17** Lengan 4 DoF ..... 50
**Gambar 3.18a** Ilustrasi putaran capit ..... 50
**Gambar 3.18b** Ilustrasi capit menutup dan capit membuka ..... 50
**Gambar 3.19** Komunikasi Serial Arduino Mega + Raspberry Pi 3 ... 52
**Gambar 3.20** Diagram alir program pada Raspberry Pi 3 ..... 54
**Gambar 3.21** Metode pengestimasian orientasi objek ..... 57
**Gambar 4.1** Hasil Perancangan Robot ..... 59
**Gambar 4.2** Grafik pengujian estimasi orientasi objek ..... 60
**Gambar 4.3** Grafik pengujian pergerakan griper ..... 64
**Gambar 4.4** Grafik Pengujian Lama Pengambilan Objek Pada Jarak 15 cm ..... 64
**Gambar 4.5** Grafik Pengujian Lama Pengambilan Objek Pada Jarak 30 cm ..... 65
**Gambar 4.6** Grafik Pengujian Lama Pengambilan Objek Pada Jarak 60 cm ..... 65

# DAFTAR TABEL

**Tabel 3.1** Data komunikasi Arduino Uno ke Raspberry Pi 2..... 44
**Tabel 3.2** Pengiriman Data Peringatan Jarak Ke Arduino Uno ..... 51
**Tabel 4.1** Presentase Kesalahan Pengujian Estimasi Orientasi Objek ..... 61
**Tabel 4.2** Presentase Kesalahan Pengujian Estimasi Orientasi Menurut Iluminansi Cahaya ..... 62
**Tabel 4.3** Tabel Pengujian Jarak Kamera dengan Objek ..... 66

*Halaman Ini Sengaja Dikosongkan*

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang

Dewasa ini, perkembangan teknologi semakin maju. Perkembangan teknologi dapat dirasakan di berbagai sektor, diantaranya: sektor industri dan sektor pemerintahan yang meliputi pula sektor keamanan, maupun sektor pelayanan masyarakat. Pada sektor keamanan, telah dirancang sistem yang mampu mempermudah manusia dalam menjaga keamanan. Misalnya, sistem keamanan rumah sudah mampu diwujudkan dengan adanya kamera CCTV. Selain itu, sistem keamanan pada kendaraan pun telah mampu diwujudkan dengan pemberian alarm.

Mengenai sektor keamanan, dapat disebut bahwa terorisme merupakan ancaman terbesar pada era sekarang ini. Merujuk pada kata terorisme, selalu tidak terlepas dari istilah bom atau pengeboman. Sehingga, tiap negara satu demi satu membentuk satuan khusus dalam menangani ancaman ini. Berhubungan dengan teror dan bom, satuan khusus ini pun memiliki tugas dalam menjinakkan bom atau bahan peledak yang menjadi ancaman dalam teror. Awalnya penjinakan bom dilakukan secara manual oleh manusia. Penjinakan dilakukan secara manual oleh orang terlatih pada satuan khusus tersebut. Namun, meskipun terlatih tetaplah besar kemungkinan bahwa bom dapat meledak sewaktu-waktu. Sehingga, risiko akan terkenanya ledakan bom sangat besar bagi penjinak bom. Sekarang ini, manusia telah mampu memodelkan robot yang mampu menjinakkan bom. Robot ini dirancang untuk mengurangi risiko jatuhnya korban saat penjinakan bom.

Mayoritas robot telah dirancang dengan kendali radio. Begitu pula dengan penelitian yang telah dilakukan di Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember pada tahun 2015 lalu. Namun, kemampuan robot tersebut masih terbatas, terutama dalam pengambilan objek. Robot belum mampu membedakan pose objek berbahaya, sehingga hanya pada pose objek tertentu saja robot dapat bekerja dengan baik. Selain itu, fitur pengambilan otomatis pun dirasa perlu ditambahkan agar kendali robot dapat dilakukan lebih mudah.

Oleh karena itu, penelitian ini dilakukan guna mengembangkan robot *mobile* yang sudah ada dengan menambahkan satu kamera lagi.

Kamera ini digunakan untuk mendeteksi pose benda berbahaya yang akan diambil. Robot diharapkan mampu dikendalikan dari jarak jauh dengan atau tanpa otomatis untuk menjelajah dan mengambil benda dalam pose apapun.

## 1.2 Perumusan Masalahan

Bedasarkan latar belakang di atas, dapat dirumuskan beberapa masalah, antara lain:

1. Bagaimana cara mendeteksi orientasi objek menggunakan kamera?
2. Bagaimana cara komunikasi serial arduino dengan Raspberry Pi 3?
3. Bagaimana robot mampu menerapkan metode autonomous pengambilan objek sesuai orientasinya?

## 1.3 Tujuan

Tujuan dari penelitian ini adalah:

1. Dapat mendeteksi posisi objek menggunakan kamera.
2. Dapat menginterpretasikan komunikasi serial arduino dengan Raspberry Pi 3.
3. Dapat menerapkan metode autonomous pengambilan objek sesuai orientasi objek.

## 1.4 Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut:

1. Robot tidak mengetahui letak halangan.
2. Robot berada pada bidang datar.
3. Objek yang didekati secara otomatis oleh robot adalah benda diam.
4. Tinggi objek 10 sampai 15 cm dan lebar benda tidak melebihi 5cm.
5. Objek berwarna merah dan terdeteksi memiliki panjang lebih dari 5cm.

## 1.5 Metodologi Penelitian

Metodologi yang digunakan pada penelitian tugas akhir ini adalah:

1. **Studi Literatur**

Pada tahap studi literatur dilakukan pengumpulan dasar teori yang menunjang dalam penulisan tugas akhir. Dasar teori tersebut diambil dari buku, jurnal ilmiah, artikel-artikel di internet dan forum-forum diskusi.

2. **Persiapan *Hardware***
   Tahap persiapan *hardware* merupakan tahap dimana sistem yang akan diprogram disiapkan. Sistem *hardware* sudah tersedia, sehingga tidak memerlukan perancangan lagi.
3. **Perancangan *Software***
   Setelah *Hardware* siap, yang perlu dilakukan adalah merancang *software*. Pada proses ini, dilakukan pemrograman kamera kinect sehingga mampu mendeteksi objek termasuk mendeteksi posisi keadaan/pose objek.
4. **Perancangan sistem**
   Pada tahap ini, *hardware* yang telah siap diisi dengan program yang telah dirancang. Diharapkan hasil dari perancangan sesuai dengan yang diinginkan.
5. **Pengujian Alat**
   Pengujian alat dilakukan untuk menentukan keandalan dari alat yang telah dirancang. Pengujian dilakukan untuk melihat apakah *software* dan *hardware* dapat bekerja secara baik. Pengujian dilakukan secara *real time*. Metode pengujian ini menggunakan kamera secara aktif. Jadi, *image* yang diproses adalah hasil penangkapan dari kamera secara langsung/*real time*.
6. **Penulisan Laporan Tugas Akhir**
   Tahap penulisan laporan tugas akhir dilakukan pada saat tahap pengujian alat dimulai serta setelahnya.

## 1.6 Sistematika Penulisan

Laporan tugas akhir ini ditulis berdasarkan format yang telah ditentukan oleh Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember. Tugas akhir ini terdiri dari lima bab yang akan dijelaskan pada poin di bawah.

- Bab 1 : Pendahuluan
  Bab ini menjelaskan latar belakang, perumusan masalah, tujuan, sistematika penulisan, metodologi, dan relevansi tugas akhir.
- Bab 2 : Dasar Teori

Bab ini menjelaskan tentang berbagai macam teori pendukung dalam penulisan tugas akhir. Bab ini meliputi dasar teori dari

- Bab 3 : Perancangan Sistem
  Bab ini menjelaskan tentang perancangan sistem, baik perangkat keras (*hardware*) ataupun perangkat lunak (*software*) untuk pengestimasian pose objek berbahaya.
- Bab 4 : Pengujian Analisis
  Bab ini menjelaskan hasil uji coba sistem beserta analisa.
- Bab 5 : Penutup
  Bab ini merupakan bab terakhir yang berisikan kesimpulan yang diperoleh dari pembuatan Tugas Akhir, serta saran-saran untuk pengembangan lebih lanjut.

## 1.7 Relevansi

Mata kuliah pendukung tugas akhir ini adalah Sistem Mikroprosesor dan Mikrokontroler, Dasar Interaksi Manusia dan Mesin, Dasar Sistem Elektronika Cerdas, dan Penginderaan Visual Elektronika. Tugas akhir ini memiliki relevansi dengan *pose detection* menggunakan kamera.

Hasil dari tugas akhir ini diharapkan dapat memberikan sumbangsih penelitian dalam bidang elektronika dan juga bidang penelitian menggunakan kamera untuk aplikasi *pose detection***.**

# BAB II

# DASAR TEORI DAN TINJAUAN PUSTAKA

Pada bab ini akan dijelaskan berbagai macam dasar teori dan tinjauan pustaka yang digunakan dalam perancangan sistem. Dalam dasar teori akan dijelaskan berbagai macam teori penunjang dalam penulisan dan perancangan tugas akhir. Sedangkan dalam tinjauan pustaka akan dijelaskan tentang sistem yang berhubungan tugas akhir ini oleh penulis-penulis sebelumnya.

## 2.1 Dasar Teori

### 2.1.1 OpenCV

OpenCV adalah suatu *open source computer vision library* yang tersedia pada *http://SourceForge.net/projects/opencvlibrary. Library* tertulis dalam bahasa C dan C++ dan dapat dioperasikan di Linux, Windows, serta Mac OS X. Terdapat pengembangan pada antarmuka untuk Python, Ruby, Matlab, dan bahasa pemrograman lainnya[3].

OpenCV didesain untuk efisiensi komputasional dengan fokus kuat pada waktu pengaplikasian. OpenCV tertulis dalam C yang dioptimisasi dan dapat mengambil kelebihan dari *multicore processor.* Salah satu tujuan OpenCV adalah menyediakan kemudahan penggunaan infrastruktur komputer vision. Diharapkan hal ini dapat membantu orang-irang dalam mengaplikasikan komputer vision dalam kehidupan. Library OpenCV terdiri dari 500 fungsi yang merentang di banyak daerah pada penglihatan(vision), termasuk inspeksi produk pabrik, *medical imaging,* keamanan, antarmuka, kalibrasi kamera, stereo vision, dan robotika. OpenCV juga berisikan *general-purpose Machine Learning Library* (MLL). *Sublibrary* ini difokuskan pada pola *recognition* dan *clustering.*

Lisensi *open source* untuk OpenCV telah distruktur, sehingga OpenCV dapat digunakan oleh siapapun. Karena lisensi liberal ini pula, terdapat komunitas user yang besar yang termasuk perusahaan besar (IBM, Microsoft, Intel, SONY, Siemens, dan Google) dan pusat penelitian (Stanford, MIT, CMU, Cambridge, dan INRIA). Di Yahoo, terdapat juga forum yang mendiskusikan masalah OpenCV dengan anggota sekitar 20.000 pengguna. OpenCV sudah terkenal di dunia,

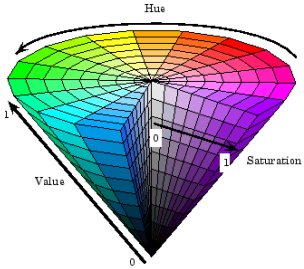
dengan komunitas yang sangat besar di China, Jepang, Russia, Eropa, dan Israel[3].

### 2.1.2 *Color Detection and Object Tracking*

Color filter adalah suatu teknik pengolahan citra yang yang dipakai untuk memanipulasi suatu citra berdasarkan warna spesifik. Cara kerjanya adalah dengan membandingkan komponen warna setiap pixel citra dengan warna spesifik. Apabila warnanya sesuai dengan warna spesifik komponen warna pixel tersebut dibiarkan saja. Namun, bila warnanya tidak sesuai dengan warna spesifik maka komponen warna pixel tersebut diubah menjadi warna background, biasanya menjadi warna hitam.

Warna yang digunakan dalam color filtering dapat direpresentasikan dalam berbagai ruang warna. Ada beberapa ruang warna yang dikenal, antara lain RGB (Red, Green, Blue), HSV (Hue, Saturation, Value), YCrCb, dan sebagainya. HSV merupakan ruang warna yang sangat cocok untuk mengidentifikasi warna-warna dasar, dimana warna dasar ini digunakan dalam penelitian sebagai warna identifikasi robot. Selain itu, HSV menoleransi terhadap perubahan intensitas cahaya. Inilah yang menjadi keunggulan HSV dibandingkan dengan ruang warna lainnya.

### 2.1.3 HSV *(Hue, Saturation, Value)*




**Gambar 2.1** Warna HSV

Warna adalah hasil persepsi dari cahaya dalam spektrum wilayah yang terlihat oleh retina mata, dan memiliki panjang gelombang antara 400nm sampai dengan 700nm. Sedangkan ruang warna adalah model matematis abstrak yang menggambarkan cara agar suatu warna dapat direpresentasikan sebagai baris angka biasanya

dengan nilai-nilai dari tiga atau empat buah warna atau komponen. contohnya adalah ruang warna RGB, ruang warna CMY/CMYK, ruang warna YIQ, ruang warna YCbCr, ruang warna HSI, HSL, HSV, ruang warna CIELAB.

Model warna HSV mendefinisikan warna dalam terminologi *Hue, Saturation* dan *Value*. Keuntungan HSV adalah terdapat warna-warna yang sama dengan yang ditangkap oleh indra manusia. Sedangkan, warna yang dibentuk model lain seperti RGB merupakan hasil campuran dari warna-warna primer.

Model HSV, pertama kali diperkenalkan oleh A.R Smith pada tahun 1978. Model warna HSV memiliki 3 karakteristik pokok, yaitu *Hue, Saturation* dan *Value*. Berikut penjelasan mengenai ketiganya:

- *Hue* : menyatakan warna sebenarnya, seperti merah, violet, dan kuning dan digunakan menentukan kemerahan (redness), kehijauan (greeness), dsb.
- *Saturation* : kadang disebut chroma, adalah kemurnian atau kekuatan warna.
- *Value* : kecerahan dari warna. Nilainya berkisar antara 0-100 %. Apabila nilainya 0 maka warnanya akan menjadi hitam, semakin besar nilai maka semakin cerah dan muncul variasi-variasi baru dari warna tersebut.

Terdapat dua cara perhitungan atau pengkonversian nilai RGB ke HSV. Cara perhitungan pertama yaitu:

$$H = \tan\left(\frac{3(G-B)}{(R-G)+(R-B)}\right) \quad (1)$$

$$S = 1 - \frac{MIN(R,G,B)}{V} \quad (2)$$

$$V = \frac{R+G+B}{3} \quad (3)$$

Kemudahan cara pertama ternyata menimbulkan permasalahan, cara pertama membuat *hue* tidak terdefinisi jika *Saturation* bernilai 0.

Solusi kedua untuk mendapatkan setiap nilai HSV adalah dengan rumus berikut :

$$r = \frac{R}{(R+G+B)}, g = \frac{G}{(R+G+B)}, b = \frac{B}{(R+G+B)}$$

$$V = \max(r, g, b)$$

$$S = \begin{cases} 0, & \text{jika V} = 0 \\ 1 - \frac{\min(r,g,b)}{V}, & \text{V} > 0 \end{cases}$$

$$H = \begin{cases} 0, & jika\, S = 0 \\ \frac{60*(g-b)}{S*V}, & jika\, V = r \\ 60 * \left[2 + \frac{b-r}{S*V}\right], & jika\, V = g \\ 60 * \left[4 + \frac{r-g}{S*V}\right], & jika\, V = b \end{cases}$$

$$H = \text{H} + 360 \text{ jika H} < 0 \quad (4)$$

### 2.1.4 Morfologi

Morfologi adalah operasi pemrosesan nilai-nilai gambar yang memroses gambar berdasarkan bentuk. Operasi morfologi menerapkan penstrukturan elemen ke gambar input, membuat output sesuai ukuran gambar input. Pada operasi morfologi, nilai tiap piksel pada gambar output merupakan perbandingan dari piksel yang sama pada gambar input dengan piksel sebelahnya. Dengan memilih ukuran dan bentuk daerah sekitar, dapat disusun operasi morfologi yang sensitif terhadap bentuk spesifik pada gambar input.

Operasi morfologi paling sederhana adalah *erosion* dan *dilation*. *Erosion* menghapus piksel pada batas objek, sedangkan *Dilation* menambah piksel ke batas objek pada gambar. Nilai piksel yang ditambahkan atau dihapus bergantung pada ukuran dan bentuk dari elemen struktur yang digunakan untuk memroses gambar. Pada morfologi *dilation* dan *erosion,* status dari tiap piksel dalam gambar output ditentukan dengan mengaplikasikan aturan pada piksel yang sama dan sebelahnya pada gambar input. Aturan digunakan untuk memroses piksel dimaksudkan pada operasi *dilation* atau *erosion*. Aturan pada *dilation* yaitu, nilai piksel output adalah nilai maksimum dari seluruh nilai piksel pada piksel input sebelahnya. Pada gambar biner, jika ada piksel yang bernilai 1, maka outputnya juga bernilai 1. Aturan pada *erosion* ialah nilai piksel output adalah nilai minimum

dari seluruh nilai piksel pada piksel input sebelahnya. Pada gambar biner, jika ada piksel yang bernilai 0, maka outputnya juga bernilai 0[2].

### 2.1.5 *Edge Detection*

*Canny edge detector* adalah operator deteksi tepi yang menggunakan algoritma multistage untuk mendeteksi range yang lebar dari tepi dalam gambar. *Canny edge detector* dikembangkan oleh John F. Canny pada 1986[2]. Canny juga menghasilkan teori komputasi dari deteksi tepi yang menjelaskan cara kerja teknik ini.

Deteksi tepi Canny adalah teknik untuk mengekstrak informasi struktural yang berguna dari benda-benda visi yang berbeda dan secara dramatis mengurangi jumlah data yang akan diproses. Ini telah banyak diterapkan di berbagai sistem visi komputer. Canny telah menemukan bahwa persyaratan untuk aplikasi deteksi tepi pada sistem visi beragam relatif sama. Dengan demikian, solusi deteksi tepi untuk mengatasi kebutuhan tersebut dapat diimplementasikan dalam berbagai situasi. Kriteria umum untuk deteksi tepi meliputi:

1. Deteksi tepi dengan tingkat kesalahan yang rendah, maksudnya deteksi harus akurat menangkap sebanyak tepi yang ditunjukkan pada gambar.
2. Titik tepi terdeteksi dari operator harus akurat melokalisasi di tengah tepi.
3. Tepi yang diberikan dalam gambar hanya ditandai sekali, dan jika mungkin, noise gambar tidak menghasilkan tepi palsu.

Untuk memenuhi persyaratan tersebut, Canny menggunakan kalkulus variasi – teknik untuk menemukan fungsi yang mengoptimalkan fungsi yang ada. Fungsi optimal dalam detektor Canny dideskripsikan oleh jumlah dari empat bilangan eksponensial, tapi ini dapat dididekati oleh turunan pertama dari Gaussian.

Di antara metode deteksi tepi yang dikembangkan sejauh ini, algoritma deteksi tepi canny adalah salah satu metode yang dapat diandalkan. Karena optimalitas untuk memenuhi dengan tiga kriteria untuk deteksi tepi dan kesederhanaan proses implementasi, hal ini membuatnya menjadi salah satu algoritma paling populer dalam deteksi tepi.

Algoritma proses dari deteksi tepi Canny dapat dijabarkan dalam lima tahap, yaitu:

1. Menggunakan gaussian filter untuk menghaluskan gambar dengan menghilangkan noise.
2. Menemukan intensitas gradien dari gambar.
3. Mengaplikasikan supresi non-maximum untuk menyingkirkan respon palsu deteksi tepi.
4. Mengaplikasikan double threshold untuk menentukan tepi potensial.
5. Melacak tepi dengan hysteresis : Finalisasi deteksi tepi dengan menekan semua tepi lain yang lemah dan tidak terhubung ke tepi kuat.

**Gambar 2.2** Deteksi Tepi Canny

### 2.1.6 Hough Transform

Hough transform adalah fitur teknik ekstraksi digunakan pada analisis gambar, *computer vision,* dan *digital image processing.* Tujuan dari teknik ini adalah untuk menemukan contoh sempurna dari objek dalam kelas bentuk tertentu oleh prosedur pemungutan suara. Prosedur voting dilakukan dalam ruang parameter, dari mana calon objek diperoleh sebagai maxima lokal dalam ruang akumulator disebut yang secara eksplisit dibangun oleh algoritma untuk menghitung transformasi Hough.

Hough transform awalnya hanya terfokus pada identifikasi garis dalam gambar, tetapi pada perkembangannya hough transform sudah mampu mendeteksi tepi pada bentuk-bentuk objek yang lain (tidak sebatas garis lurus) seperti elips maupun lingkaran. Transformasi Hough yang digunakan saat ini diciptakan oleh Richard Duda dan Peter Hart pada tahun 1972 , yang menyebutnya sebagai “generalized Hough transform”[2]. Transformasi itu dipopulerkan di masyarakat visi komputer oleh Dana H. Ballard melalui sebuah artikel jurnal 1981 berjudul “Generalizing the Hough transform to detect arbitrary shapes”.

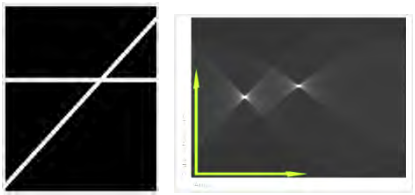
Transformasi Hough memiliki beberapa perbedaan rumus yang diterapkan. Semuanya tergantung pada jenis objek yang dicari, misalnya untuk mencari objek garis akan digunakan fungsi garis seperti berikut ini:

$$x \cos \Theta + y \sin \Theta = r \quad (5)$$

Dengan x dan y merupakan titik koordinat yang menyusun objek garis tersebut, sedangkan teta adalah sudut yang dibentuk antara objek garis dengan sumbu x, dan r merupakan jarak antara garis dengan titik pusat (0,0).

Jika objek yang dicari berupa lingkaran, maka digunakan transformasi lingkaran Hough. Prosedur yang digunakan dalam mendeteksi lingkaran adalah sama dengan transformasi Hough pada objek garis, tapi dikerjakan pada ruang dimensi yang lebih kompleks, yaitu dalam parameter ruang 3D (Xo,Yo,R). Di mana Xo dan Yo merupakan koordinat pusat lingkaran dan r adalah jari-jari lingkaran seperti persamaan berikut:

$$(x-x_0)^2 + (y-y_0)^2 = r^2 \quad (6)$$



**Gambar 2.3** Gambar Input dan Hasil Pemetaan

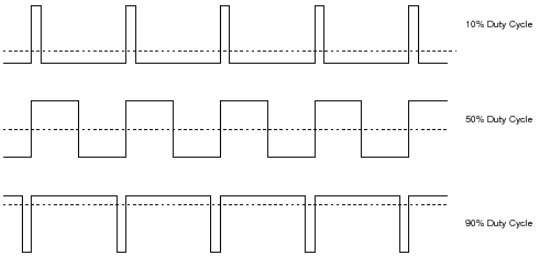
Secara umum Transformasi Hough bekerja dengan memanfaatkan sebuah deret Array yang dinamakan akumulator. Array akumulator ini memiliki dimensi yang berbeda-beda tergantung dari jumlah parameter dari objek yang dicari. Misalnya, pada transformasi garis Hough memerlukan 2 buah parameter yakni r dan teta, maka dibentuklah sebuah deret array akumulator yang berdimensi dua. Pencarian kemudian dilakukan terhadap sebuah area pixel dengan mencari kemungkinan hubungan atau garis yang ada. Setiap kemungkinan hubungan garis dihitung nilai r dan tetanya terhadap titik pusat. Selanjutnya, menyimpan nilai r dan tetanya dari setiap kemungkinan hubungan tersebut pada array akumulator. Nilai-nilai pada akumulator akan dipetakan ke dalam sebuah grafik yang

dinamakan grafik akumulator dengan teta sebagai absis dan r sebagai ordinat.

### 2.1.7 Pulse Width Modulation

PWM adalah suatu sinyal yang dikirim dengan frekuensi tetap namun dapat memiliki panjang pulse yang berbeda-beda dalam setiap periodenya. Perbedaan ini biasanya disebut dengan *Duty Cycle*, yaitu perbandingan lama pulse dengan keseluruhan periode sinyal. PWM adalah istilah yang biasanya disebut sebagai sinyal keluaran (*output*) analog [4]. Caranya adalah dengan mengubah tegangan rata-rata setiap periodenya. PWM dengan kondisi tertentu dapat dikonversi menjadi Pulse Position modulation mengingat karakteristiknya yang sama.

Pulse Position modulation lebih sering dikenal dengan PPM. PPM banyak digunakan dalam sinyal kontrol motor servo. PPM biasanya di kaitkan dengan Pulse Width Modullation atau PWM. PPM memiliki sinyal high yang sama, namun karena dimungkinkan jarak antar sinyal high berbeda-beda, maka PPM tidak memiliki frekuensi tetap. Jarak antar pulse (interval) yang menentukan data yang dibawa[5].




**Gambar 2.4** Sinyal PWM

Penelitian ini menggunakan PWM sebagai sinyal kontrol ESC dan juga motor servo. PWM sangat banyak ditemukan sebagai pengontrol kecepatan motor DC. PWM mengatur kecepatan motor DC menggunakan tegangan rata-rata output dan frekuensi tertentu yang telah disesuaikan. Akan tetapi, dengan menggunakan PWM yang ditambahkan ESC maka penggunakan frekuensi PWM untuk motor DC dan motor servo dapat disamakan.

### 2.1.8 Motor Servo

**Gambar 2.5** Motor Servo

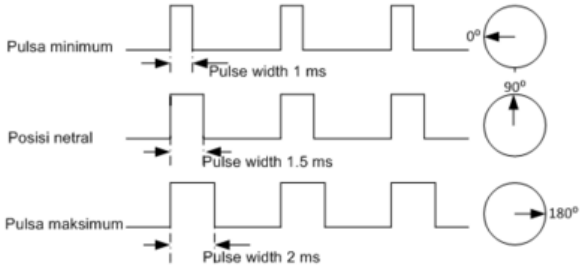
Motor servo adalah sebuah perangkat atau aktuator putar (motor) yang dirancang dengan sistem kontrol umpan balik loop tertutup (servo), sehingga dapat di set-up atau di atur untuk menentukan dan memastikan posisi sudut dari poros output motor. motor servo merupakan perangkat yang terdiri dari motor DC, serangkaian gear, rangkaian kontrol dan potensiometer. Serangkaian gear yang melekat pada poros motor DC akan memperlambat putaran poros dan meningkatkan torsi motor servo, sedangkan potensiometer dengan perubahan resistansinya saat motor berputar berfungsi sebagai penentu batas posisi putaran poros motor servo.

Penggunaan sistem kontrol loop tertutup pada motor servo berguna untuk mengontrol gerakan dan posisi akhir dari poros motor servo. Penjelasan sederhananya begini, posisi poros output akan di sensor untuk mengetahui posisi poros sudah tepat seperti yang di inginkan atau belum, dan jika belum, maka kontrol input akan mengirim sinyal kendali untuk membuat posisi poros tersebut tepat pada posisi yang diinginkan. Untuk lebih jelasnya mengenai sistem kontrol loop tertutup, perhatikan contoh sederhana beberapa aplikasi lain dari sistem kontrol loop tertutup, seperti penyetelan suhu pada AC, kulkas, setrika dan lain sebagainya.

Motor servo biasa digunakan dalam aplikasi-aplikasi di industri. Selain itu juga digunakan dalam berbagai aplikasi lain seperti pada mobil mainan radio kontrol, robot, pesawat, dan lain sebagainya. Ada dua jenis motor servo, yaitu motor servo AC dan DC. Motor servo AC lebih dapat menangani arus yang tinggi atau beban berat, sehingga sering diaplikasikan pada mesin-mesin industri. Sedangkan

motor servo DC biasanya lebih cocok untuk digunakan pada aplikasi-aplikasi yang lebih kecil.
Bila dibedakan menurut rotasinya, umumnya terdapat dua jenis motor servo yang dan terdapat di pasaran, yaitu motor servo rotation $180^0$ dan servo rotation continuous. Motor servo standard (servo rotation $180^0$) adalah jenis yang paling umum dari motor servo, dimana putaran poros outputnya terbatas hanya $90^0$ kearah kanan dan $90^0$ kearah kiri. Dengan kata lain total putarannya hanya setengah lingkaran atau $180^0$. Motor servo rotation continuous merupakan jenis motor servo yang sebenarnya sama dengan jenis servo standard, hanya saja perputaran porosnya tanpa batasan atau dengan kata lain dapat berputar terus, baik ke arah kanan maupun kiri.

Motor servo dikendalikan dengan memberikan sinyal modulasi lebar pulsa (Pulse Wide Modulation / PWM) melalui kabel kontrol. Lebar pulsa sinyal kontrol yang diberikan akan menentukan posisi sudut putaran dari poros motor servo. Sebagai contoh, lebar pulsa dengan waktu 1,5 ms (mili detik) akan memutar poros motor servo ke posisi sudut $90^0$. Bila pulsa lebih pendek dari 1,5 ms maka akan berputar ke arah posisi $0^0$ atau ke kiri (berlawanan dengan arah jarum jam), sedangkan bila pulsa yang diberikan lebih lama dari 1,5 ms maka poros motor servo akan berputar ke arah posisi $180^0$ atau ke kanan (searah jarum jam). Lebih jelasnya perhatikan gambar dibawah ini.




**Gambar 2.6** Pulsa dan pergerakan motor servo

Ketika lebar pulsa kendali telah diberikan, maka poros motor servo akan bergerak atau berputar ke posisi yang telah diperintahkan, dan berhenti pada posisi tersebut dan akan tetap bertahan pada posisi tersebut. Jika ada kekuatan eksternal yang mencoba memutar atau

mengubah posisi tersebut, maka motor servo akan mencoba menahan atau melawan dengan besarnya kekuatan torsi yang dimilikinya (rating torsi servo). Namun motor servo tidak akan mempertahankan posisinya untuk selamanya, sinyal lebar pulsa kendali harus diulang setiap 20 ms (mili detik) untuk menginstruksikan agar posisi poros motor servo tetap bertahan pada posisinya.

### 2.1.9 Komunikasi Serial

Komunikasi serial adalah komunikasi dengan pengiriman data per-bit secara berurutan dan bergantian. Komunikasi ini memiliki kelebihan yaitu hanya membutuhkan satu jalur dan kabel yang sedikit dibandingkan dengan komunikasi paralel. Pada prinsipnya, komunikasi serial merupakan salah satu metode komunikasi data di mana hanya satu bit data yang dikirimkan melalui seuntai kabel pada suatu waktu tertentu. Komunikasi serial adalah kasus khusus komunikasi paralel dengan nilai n = 1, atau dengan kata lain adalah suatu bentuk komunikasi paralel dengan jumlah kabel hanya satu dan hanya mengirimkan satu bit data secara simultan. Hal ini dapat disandingkan dengan komunikasi paralel yang sesungguhnya di mana n-bit data dikirimkan bersamaan, dengan nilai umumnya $8 \leq n \leq 128$.

Komunikasi serial ada dua macam, serial asinkron dan serial sinkron. Serial sinkron adalah komunikasi dimana hanya ada satu pihak (pengirim atau penerima) yang menghasilkan clock dan mengirimkan clock tersebut bersama-sama dengan data. Contoh pengunaan serial sinkron terdapat pada transmisi data keyboard. Serial asinkron adalah komunikasi dimana kedua pihak (pengirim dan penerima) masing-masing menghasilkan clock, namun hanya data yang ditransmisikan, tanpa clock. Agar data yang dikirim sama dengan data yang diterima, maka kedua frekuensi clock harus sama dan harus terdapat sinkronisasi. Setelah adanya sinkronisasi, pengirim akan mengirimkan datanya sesuai dengan frekuensi clock pengirim dan penerima akan membaca data sesuai dengan frekuensi clock penerima. Contoh penggunaan serial asinkron adalah pada Universal Asynchronous Receiver Transmitter (UART) yang digunakan pada serial port (COM) komputer.

Antarmuka kanal serial lebih kompleks dibandingkan dengan antarmuka melalui kanal paralel, hal ini disebabkan karena adanya proses konversi data pararel menjadi serial atau sebaliknya menggunakan piranti tambahan yang disebut UART (Universal

Asynchronous Receiver/Transmitte) dan lebih banyak register yang digunakan atau terlibat.

Namun, di sisi lain antarmuka kanal serial memiliki beberapa kelebihan dibandingkan secara paralel, antara lain:

1. Kabel untuk komunikasi serial bisa lebih panjang dibandingkan dengan paralel.
2. Jumlah kabel lebih sedikit.
3. Kebanyakan piranti saat ini (palmtop, organizer, hand-phone dan lain-lain) menggunakan teknologi infra merah untuk komunikasi data.
4. Untuk teknologi *embedded system*, banyak mikrokontroler yang dilengkapi dengan komunikasi serial (baik seri RISC maupun CISC) atau Serial Communication Interface (SCI).

### 2.1.10 Sensor Ultrasonik

Sensor ultrasonik didunia pasaran elektronik juga disebut dengan Ping sensor atau Paralax sensor. Sensor ultrasonic adalah sensor yang memancarkan sinyal ultrasonic kemunian menangkap kembali sinyal tersebut. Sensor ultrasonic dapat digunakanu untuk menghitung jarak dengan mengukur jeda waktu antara sinyal saat dikirim dan di terima.

Pada tugas akhir ini sensor ultrasonic digunakan untuk menghitung jarak robot dengan halangan didepan robot. Jarak dapat dihitung menggunakan perhitungan sebagai berikut.

$$S = \frac{V * t}{2} \tag{7}$$

Keterangan :

S = Jarak antara sensor dengan halangan dihitung dalam satuan centimeter.

V=29 cm/s, yaitu kecepatan suara diudara.

t= Waktu yang ditempuh mulai dari sinyal dikirim sampai dengan sinyal kembali dihitung dalam satuan microsecond

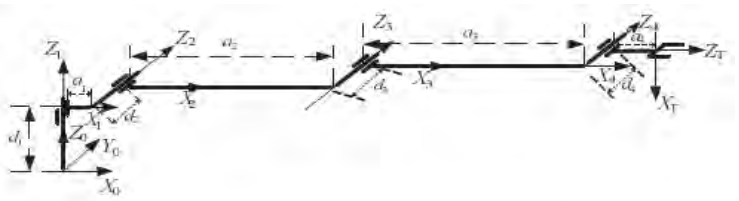
### 2.1.11 Robot Kinematics

Kinematika adalah ilmu yang mempelajari gerak tanpa mengindahkan gaya yang menyebabkan gerakan tersebut [6]. *Robot kinematics* adalah suatu disiplin ilmu yang mengaplikasikan geometri untuk mempelajari gerakan suatu titik yang menyusun struktur sistem robot. Geometri yang dimaksud adalah setiap bagian robot dimodelkan

sebagai benda tegar dan setiap sambungan bagian robot menghasilkan gerakan translasi atau rotasi secara murni. Benda tegar adalah suatu kesatuan benda secara ideal yang bentuk dan ukurannya tidak berubah ketika mendapatkan suatu gaya tertentu. Setiap bagian robot secara teori dianggap sebagai benda tegar, meskipun pada kenyatannya tidaklah sama. *Robot kinematics* mempelajari hubungan antara dimensi dan setiap sambungan (*joint*) dari beberapa bagian kinematika dan posisi, kecepatan dan akselerasi dari setiap bagian robot untuk merencanakan kontrol pergerakan dan menghitung gaya dan torsi aktuator.

Robot terdiri dari link yang dihubungkan oleh joint. Biasanya setiap joint dilengkapi dengan instrument untuk mengukur posisinya relatif terhadap posisi bagian yang diukur. Pada joint yang berputar perpindahan poisi link disebut dengan sudut joint (*joint angles*). Beberapa robot juga terdiri dari sambungan yang bergerak secara translasi tau prismatic. Perpindahan gerakan pada joint yang seperti ini biasanya disebut *joint offset*. Setiap bagian joint akan menghasilkan tingkat kebebasan robot dalam bergerak (*Degree of Freedom (Dof)*)[7].

Alat yang sangat fundamental pada kinematika robot adalah persamaan kinematika dari rantai kinematika yang menyusun robot. Persamaan kinematika juga digunakan pada biomekanik dan juga animasi komputer. Ada dua persamaan utama yang biasa digunakan pada kinematika robot, yaitu *Forward kinematics* (Kinematika Maju) dan *Reverse kinematics* (Kinematika Mundur). *Forward kinematics* menggunakan persamaan kinematika untuk menghitung posisi ujung (*end-effector*) dari nilai yang telah ditentukan pada setiap parameter sambungan. Proses sebaliknya, yaitu menghitung posisi *joint* untuk menentukan posisi *end-effector* disebut *Reverse kinematics*. Dimensi robot dan persamaan kinematika menentukan volum jangkauan robot yang biasa disebut *workspace* [7].



**Gambar 2.7** Lengan 4 Dof

### 2.1.12 USB to TTL

USB to TTL merupakan divais yang digunakan untuk memudahkan komunikasi serial antar mikrokontroler dengan mikrokontroler lainnya. Dalam aplikasinya, USB to TTL biasa dipakai dengan perantara kabel. Komunikasi serial USB to TTL memanfaatkan komunikasi antara USB dan serial UART. Terdapat 5 pin pada USB to TTL, yaitu GND, TX, RX, 5V, dan 3.3V. GND adalah *ground*. Pin GND ini difungsikan agar tiap mikrokontroler yang tersambung memiliki keterhubungan dalam satu jalur dengan satu *ground*. Pin TX merupakan pin yang digunakan untuk mengirimkan (transmit) data dari mikrokontroler A ke mikrokontroler B. Pin RX merupakan pin yang memiliki fungsi berkebalikan dengan pin TX, yaitu menerima (*receive*) data. Dua pin sisa merupakan pin tegangan suplai. Pin ini digunakan untuk menyuplai mikrkontroler yang saling terhubung/menghubungkan suplai pada mikro A ke mikro B. Penggunaannya sendiri bisa dipilih salah satu diantara kedua nilai yang tertera, yaitu 5V dan 3.3V.

**Gambar 2.8** USB to TTL

## 2.2 Tinjauan Pustaka

### 2.2.1 Pustaka Pendukung

Penggunaan *color and edge detection* pada sistem vision suatu *mobile robot* telah diwujudkan oleh Efthimia Kafaela[8]. Dua segmen ini digunakan untuk lokalisasi objek. Sistem vision dengan *color and edge detection* dianggap memiliki kelebihan pada keandalan, fleksibilitas, dan kesukarannya. Pada *color and edge detection,* digunakan analisa ruang warna untuk mengetahui nilai tiap piksel dari gambar yang di dapatkan. Analisa ini dilakukan berdasarkan nilai dari hue dari HSV gambar yang terekam. Dengan menggunakan hue,

warna yang diterima jauh lebih stabil dibandingkan dengan menggunakan gambar RGB (*Red, Green, Blue)* biasa. Gambar RGB lebih mudah terpengaruh cahaya atau iluminansi dari pada gambar HSV (*Hue, Saturation, Value).*

Menurut Simardeep Kaur dan Dr. Vijay Kumar Banga, pada penelitiannya tentang RGB dan HSV, gambar HSV memiliki akurasi yang lebih tinggi dibandingkan dengan gambar RGB[9]. Akurasi yang dimaksud adalah nilai rata-rata dari *Recall* dan Presisi. *Recall* ialah mengukur kemampuan sistem untuk mengambil semua model yang relevan, sementara presisi ialah mengukur kemampuan sistem untuk mengambil hanya model yang relevan.

Recall = Jumlah gambar relevan yang diambil/Jumlah gambar relevan pada database.

Precision = Jumlah gambar relevan yang diambil /Jumlah seluruh gambar yang diambil.

Gambar RGB tidak memberikan informasi yang tepat ketika terdapat efek *luminance*. HSV memiliki informasi warna berupa *Hue* yaitu kedalaman warna, *Saturation* yaitu kemurnian warna, dan intensitas dari *Value* yaitu kecerahan. *Hue* mengacu pada warna merah, biru, dan kuning dengan nilai 0-360. *Saturation* dimaksudkan pada kemurnian warna dengan nilai antara 0 sampai 100%. *Value* mengacu pada kecerahan warna dan menyediakan gagasan akromatik dari warna. Pada gambar dengan YCbCr terdapat komponen *luminance, blue minus luminance, red minus luminance.* Gambar YCbCr ini harusnya lebih mudah dimanfaatkan, namun karena beberapa alasan akurasi dari YCbCr lebih rendah. Hal ini dinyatakan S.Chitra dan G. Balakrishnan dalam penelitiannya[10].

Operasi morfologi merupakan hal yang penting, apalagi pada *image processing.* Morfologi merupakan metode matematika yang sangat berguna. Morfologi memiliki banyak aplikasi, diantaranya deteksi tepi dan ekstraksi gambar. Pada penelitian yang dilakukan Harendra, morfologi digunakan untuk memberikan solusi pada korupsi, oklusi, ekspresi wajah yang berbeda, dan pose wajah yang ditangkap oleh kamera[18]. Pada fase ini, morfologi difungsikan sebagai penghilang *noise* dan penghalus gambar.

Morfologi juga digunakan oleh Fattah pada penelitiannya tentang pemisahan teks dan gambar. Penelitian ini dilakukan pada 150 majalah berbeda. Hasil yang didapatkan pada penelitian ini sangat memuaskan dan menjanjikan, karena sistem mampu memisahkan teks dan gambar[19].

Morfologi sebagai penghilang *noise*/gangguan dibuktikan pada penelitian yang dilakukan oleh Nursuriati[20]. Pada penelitiannya, dilakukan percobaan operasi morfologi pada 25 gambar. Dengan menggunakan metode morfologi yang berbeda-beda (*dilation, erotion, open, close, fill,* dan *majority).* Hasil dari penelitian menunjukkan bahwa penggunaan fungsi *erode-dilate* jauh lebih efektif untuk menghilangkan *noise* gambar. Selain itu, gambar biner pun dapat menjadi lebar saat diberikan fungsi *majority-close*.

*Hough transform* merupakan metode yang efektif untuk mendeteksi garis yang tipis pada gambar SAR (*Syntethic Aperture Radar*). Hal ini ditunjukkan oleh Jacqui Skingley dan A. J. Rye pada penelitian mereka. Pada penelitian tersebut, ditujukan untuk bermacam *post processing problem,* termasuk pada deteksi puncak dan palung pada ruang transform/perubahan[13]. Garis-garis tipis atau kurang jelas pun dapat dideteksi dengan *hough transform*.

Hough transform juga dapat digunakan untuk *general curve fitting,* dan memberikan interpretasi alternatif. Hal ini dijelaskan pada penelitian yang telah dilakukan Richard O. Duta dan Peter E. Hart[14]. Penelitian mereka fokus pada penggunaan *hough transformation* pada deteksi garis dan kurva.

Pada penelitian yang dilakukan F. Chenavier, dkk telah digunakan sistem satu kamera pada *mobile robot*[11]. Kamera pada *mobile robot* digunakan untuk mengestimasi petunjuk (*landmark)* dengan aktuator berupa servo yang diintegrasi dengan kamera, sehingga posisi kamera mampu beradaptasi dengan *landmark* yang terdeteksi. Integrasi dengan *mobile robot* pun dilakukan agar robot tetap beradaptasi dengan *landmark* tersebut.

Menurut penelitian yang dilakukan Bin Liu, *machine vision* berbasis *image processing,* pengenalan pola, teknologi komputer dan fisiologi, mengeksploitasi simulasi komputer seperti otak manusia[]. Maksudnya, komputer dapat berpikir, mencoba memahami, maupun mengekstrak informasi visual. Hal ini membuat komputer sangat berarti bagi keberlangsungan kecerdasan artifisial.

Penerapan *closed loop* pada pengenalan objek memberikan hasil yang lebih baik dibandingkan *open loop.* Hal ini ditunjukkan pada penelitian yang dilakukan . Menggunakan elemen *hue* untuk mengenali objek berdasarkan warnanya, dan menerapkan sistem *close loop* memberikan hasil yang jauh lebih baik pada robot. Hasil yang didapat dari penelitian tersebut memiliki nilai efektifitas yang tinggi pada pengenalan warna objek[16].

Pengaturan pergerakan *gripper* berdasarkan informasi dari sensor telah direalisasikan oleh Fischer lewat penelitiannya[17]. Pada penelitiannya, sensor memberikan informasi mengenai objek yang akan diambil pada *gripper*. Sehingga *gripper* mampu bergerak menyesuaikan informasi tersebut.

Penelitian *mobile robot* dengan *landmark* juga dilakukan oleh Margrit Betke dan Leonid Gurvits[12]. Pada penelitian mereka, mereka memfokuskan pada bagian *position estimator*. Didapatkan hasil bahwa algoritmanya mampu mengestimasikan posisi dan orientasi yang cukup dekat dengan posisi aktual robot.

Estimasi orientasi objek telah dilakukan Chong Chen dan Dan Schonfeld[21]. Pada penelitiannya, digunakan persamaan Silvester dengan berdasarkan *scale invariant feature transform.* Penelitian yang dilakukan mampu mengestimasi orientasi objek secara 2D.

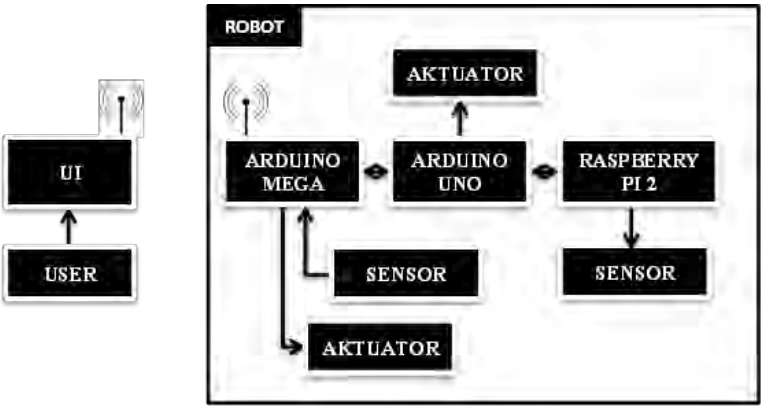
Pada penelitian yang dilakukan Jorge S. Marques, estimasi orientasi dilakukan dengan *hough transform* dan *least square*[22]. Pada penggunaan *hough transform,* ditunjukkan bahwa hasil estimasi orientasi dipengaruhi oleh lokasi fitur objek. Disimpulkan bahwa semakin dekat objek, semakin tinggi pula nilai keberhasilan estimasi orientasi.

Penggunaan komunikasi antara kamera dan motor servo ini menginisiasi dalam komunikasi antara kamera dengan lengan pada *mobile robot* yang dirancang. Lengan robot yang tentunya menggunakan motor servo sebagai komponen penyangganya, tentu dapat disangkut pautkan dengan komunikasi antara kamera dan motor servo. Sehingga pergerakan diharapkan tetap terintegrasi dalam satu sistem.

Pada penelitian James R. Allard, dijelaskan bahwa robot memerlukan *user interface* (UI) untuk bisa mengoperasikannya[15]. Pada penelitiannya, James R. Allard menyebutkan bahwa dibutuhkan suatu UI yang intuitif yang dapat mengontrol robot. Masa ini, masyarakat telah mengenal adanya *radio control. Radio control*

merupakan salah satu alat komunikasi digital. Menurut Widdy Hijriyanthi, dalam makalahnya menyebutkan bahwa komunikasi digital memiliki kelebihan, salah satunya adalah intuitif.

### 2.2.2 Sistem Awal

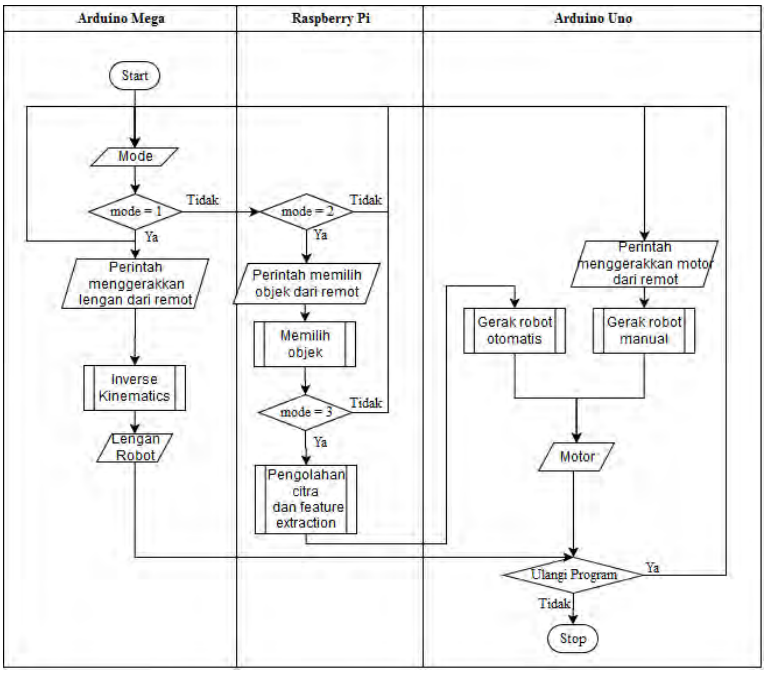



**Gambar 2.13** Diagram Sistem Awal

Pada penelitian yang lalu, telah dirancang robot pengambil benda berbahaya dengan antarmuka dan tiga mikrokontroler sebagai unit proses. Robot dirancang dengan tiga unit proses, yaitu Arduino Mega, Arduino Uno, dan Raspberry Pi 2. Pada sistem ini terdapat tiga mode yang dapat diinputkan oleh pengguna melalui antarmuka berupa radio kontrol. Tiga mode tersebut yaitu mode manual, mode memilih objek, dan mode mendekati objek. Sinyal yang dikirimkan oleh radio kontrol ini diterima oleh *receiver* yang terhubung dengan Arduino Mega. Diagram alir dari robot dari sistem yang lalu dapat dilihat pada gambar 2.10.

Pada sistem lalu terdapat dua buah kamera, yaitu kamera analog buatan AOMWAY atau bisa disebut juga kamera FPV dan kamera Logitech C170. Kamera FPV digunakan untuk memudahkan navigasi sedangkan kamera webcam digunakan untuk pengolahan citra. Pada sistem terdapat IC multiplekser yang digunakan untuk memilih output yang akan dipilih dari dua input. Kedua jenis kamera tadi

disambungkan ke IC multiplekser 4051 untuk dipilih citra mana yang akan ditampilkan pada layar monitor.




**Gambar 2.14** Diagram Alir Sistem Awal [1]

Secara garis besar, cara kerja sistem ini adalah pengguna awalnya mencari letak bom dengan memanfaatkan kamera navigasi yang terdapat pada robot dengan melihat hasil tangkapan gambarnya pada layar monitor. Cara tersebut menggunakan mode manual. Setelah objek yang akan diambil ditemukan, pengguna harus mengatur posisi robot agar tegak lurus dengan benda, sehingga proses mendekati objek dan mengambil otomatis dapat dilakukan. Setelah posisi robot sudah memenuhi, dapat dilakukan pemilihan objek dengan pengaktifan mode kedua. Pengaktifan mode kedua memanfaatkan Raspberry Pi 2 untuk melakukan proses pengolahan citra. Setelah objek dipilih, dapat diaktifkan mode ketiga untuk mendekati objek. Selain mode ketiga, dapat pula ditambahkan proses mengambil otomatis, sehingga robot

mampu mengambil objek yang sudah didekati dengan posisi tegak lurus.

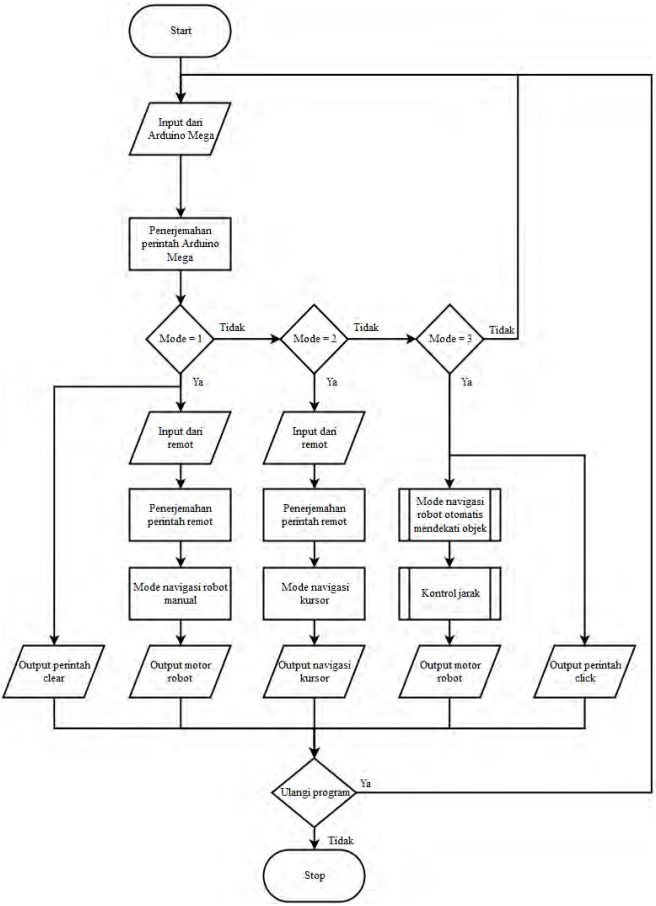
Antarmuka yang digunakan berupa radio kontrol dan layar lcd monitor, dengan Arduino Mega digunakan sebagai pusat pemrosesan. Arduino Mega memiliki aktuator berupa servo, baik itu servo pada lengan ataupun servo pada kamera navigasi. Selain itu, Arduino Mega juga mengakuisisi data dari sensor ultrasonik yang digunakan saat mode mendekati objek dihidupkan. Kelemahan dari sistem yang telah ada adalah robot hanya mampu mengambil objek dalam orientasi tegak lurus dengan robot. Sehingga jika terdapat objek yang terletak di depan robot dengan orientasi sudut tidak tegak lurus dengan robot, robot akan mengambil dengan pergerakan gripper sama seperti saat objek terletak tegak lurus dengan robot.

#### *2.2.2.1 Arduino Uno*

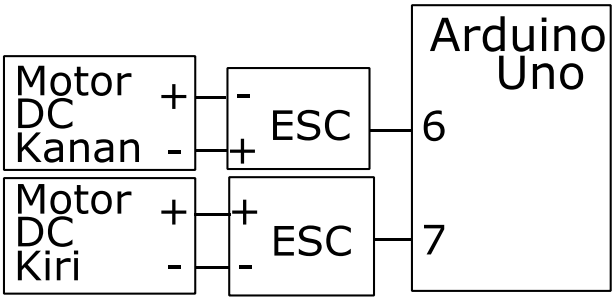
Arduino Uno digunakan untuk mengontrol laju kecepatan robot. Pengendalian laju kecepatan robot tidak dibuat satu sistem dengan Arduino Mega karena dapat mengganggu performa Arduino Mega. Gambar diagram alir program pada Arduino Uno dapat dilihat pada gambar 2.11. Arduino Uno memiliki lima output, meskipun secara perangkat keras hanya terdapat dua output. Output motor robot memiliki perangkat keras yang sama. Kedua adalah unit proses Raspberry Pi 2 yang mengoperasikan output perintah klik, output perintah *clear*, dan juga output navigasi kursor dari Arduino Uno.

Pada gambar 2.12 dapat dilihat ilustrasi pengkabelan antara Arduino Uno dengan ESC. Masing-masing pin 6 dan 7 (PWM) dari Arduino Uno dihubungkan pada ESC yang terhubung pada motor DC. Pengkabelan ini digunakan untuk mengirimkan sinyal dari Arduino Uno ke ESC guna menggerakkan Motor DC dari robot. Sehingga robot mampu berpindah satu titik ke titik lain.

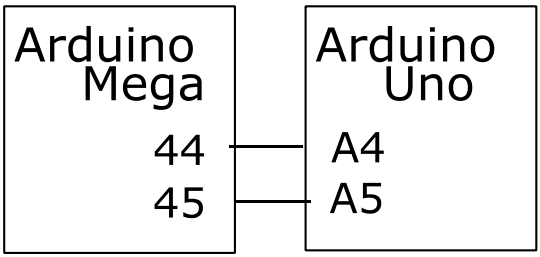
Gambar 2.13 adalah ilustrasi pengkabelan komunikasi antara Arduino Mega dan Arduino Uno. Pin 44 dan 45 pada Arduino Mega dihubungkan masing-masing dengan pin A4 dan A5 (analog input) pada Arduino Uno. Gambar ini menunjukkan bagaimana Arduino Uno dan Arduino Mega saling berkomunikasi. Komunikasi yang dimaksud ialah komunikasi agar Arduino Uno mengetahui input mode yang digunakan.




**Gambar 2.14** Diagram Alir Program Arduino Uno[1]



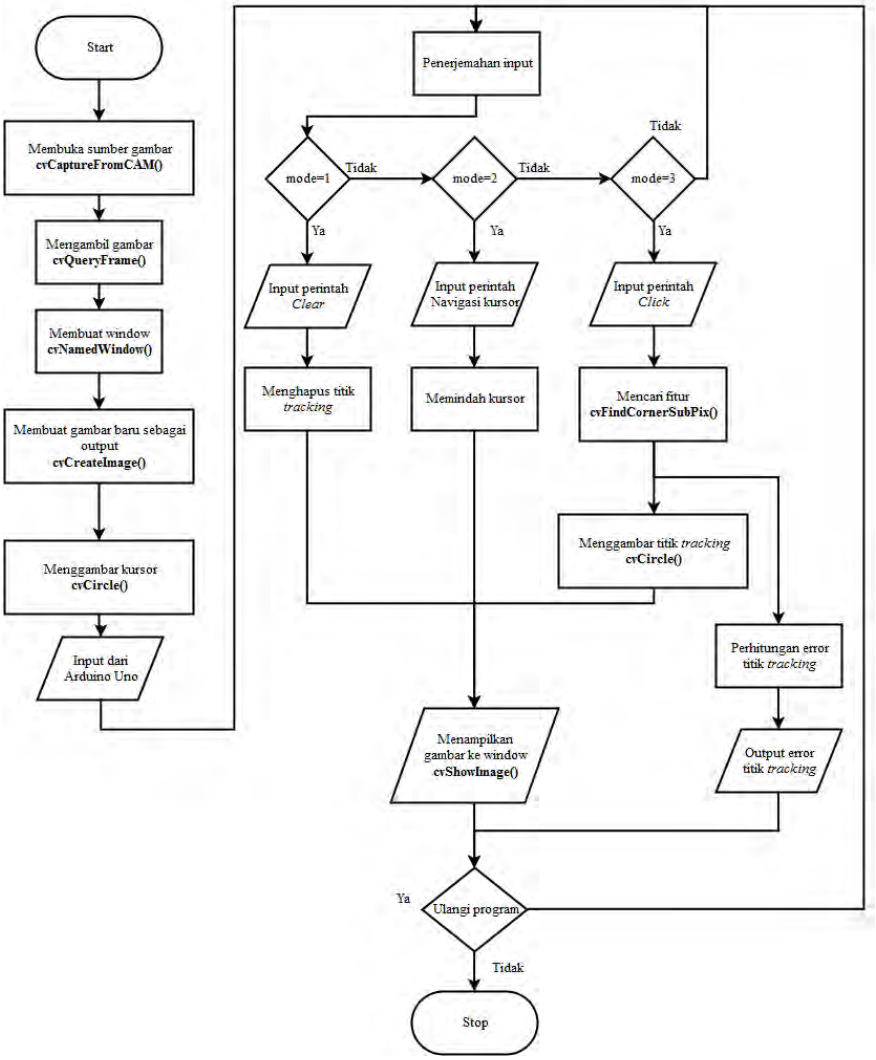

**Gambar 2.15** Pengkabelan ESC dengan Arduino Uno dan Motor[1]



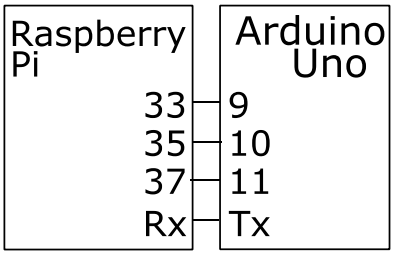

**Gambar 2.16** Pengkabelan Arduino Uno dengan Arduino Mega[1]

***2.2.2.2 Raspberry Pi 2***

Raspberry Pi 2 digunakan untuk pengolahan citra. Raspberry Pi 2 melakukan pengolahan citra dengan menggunakan *lucas kanade* pada OpenCV. Raspberry Pi 2 berkomunikasi serial dengan Arduino Uno. Pada gambar 2.13 dapat dilihat bahwa pin 33, 35, 37, dan Rx(Receiver) dari Raspberry Pi 2 dihubungkan masing-masing pada pin PWM (9, 10, 11), dan Tx(Transmitter) dari Arduino Uno. Gambar tersebut menunjukkan hubungan dari Raspberry Pi2 dengan Arduino Uno. Komunikasi dari Arduino berisi perintah untuk menggerakkan pointer saat memilih object untuk di dekati. Raspberry Pi 2 melakukan komunikasi data dengan Arduino Uno. Raspberry Pi menerima input perintah *click*, perintah *clear*, dan juga perintah navigasi kursor yang disingkat dalam tiga mode. Mode pada Raspberry Pi adalah hasil dari penerjemahan input yang berasal dari Arduino Uno.




**Gambar 2.17** Diagram Alir Raspberry Pi 2




**Gambar 2.18** Pengkabelan Arduino Uno dengan Raspberry Pi2

**Gambar 2.19** Gambar kursor (lingkaran merah) [1]

Gambar di atas adalah hasil dari pengolahan citra dengan *lucas kanade*. Gambar lingkaran ini yang difungsikan sebagai kursor. Kursor digunakan sebagai pemilih objek, sehingga mode kedua dapat dijalankan.

Raspberry Pi 2 menerjemahkan input mode dari Arduino Uno. Secara garis besar input Arduino Uno dibagi menjadi tiga mode. Pertama adalah mode *clear*, yaitu membersihkan gambar dati titik *tracking*. Mode kedua adalah mode navigasi kursor. Sedangkan yang ketiga adalah mode *click*. Perintah navigasi kursor didapatkan dari

Arduino Uno. Perintah navigasi kursor menggeser kursor keatas atau kebawah dan juga ke kanan atau ke kiri.

Mode berikutnya adalah mode ketiga atau perintah *click*. Pada saat perintah ini dimasukkan, program pada Raspberry Pi melakukan dua proses, yaitu menandai titik *tracking* dan memanggil fungsi cvFindCornerSubPix(). Fungsi ini mencari fitur terdekat dengan kursor, yaitu berjarak 10x10 pixel. Fitur pada gambar adalah bagian gambar yang unik dan dapat ditemukan setiap urutan *frame* kamera. Fungsi ini menghasilkan titik fitur. Titik fitur ini akan terus ada mengikuti benda dimana titik itu berada. Dengan demikian, titik fitur ini dapat digunakan untuk *tracking*. Dengan penambahan perhitungan nilai eror, robot akan mendekati objek dengan mengikuti titik fitur yang terdeteksi.

Mode yang terakhir adalah mode yang pertama yaitu perintah *clear*. Pada mode ketiga atau perintah *click* telah disebutkan bahwa setelah perintah click terjadi program akan menandai titik untuk *tracking*. Pada mode ini seluruh titik fitur atau titik *tracking* dihapus.

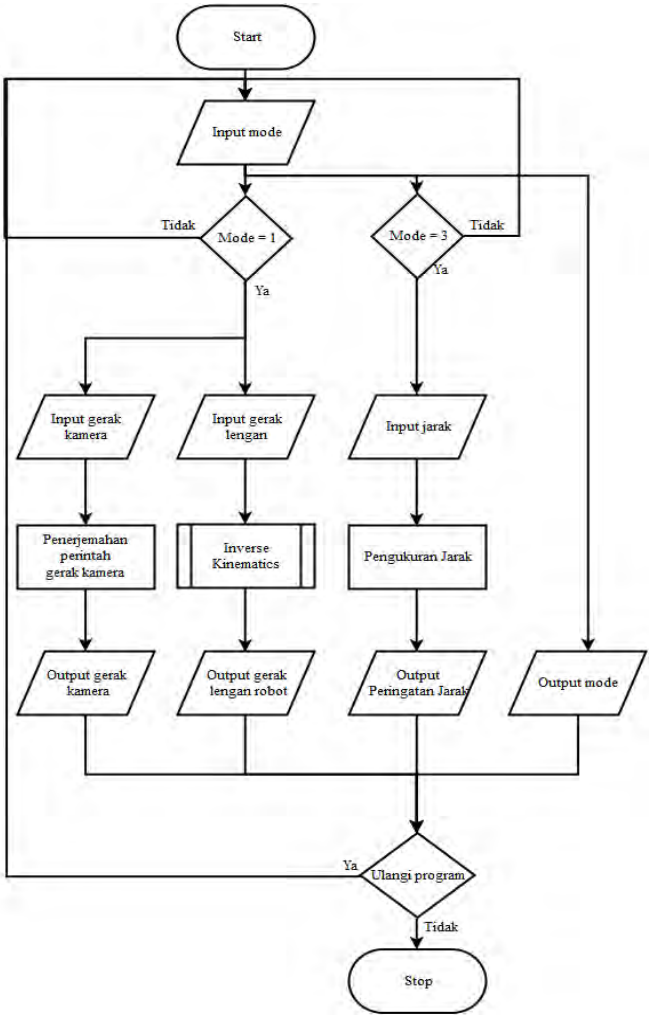
#### *2.2.2.3 Arduino Mega*

Arduino Mega digunakan utntuk mengontrol gerakan lengan robot. Pada arduino mega terdapat program inverse kinematic lengan, sehingga dapat memudahkan pengguna dalam mengontrol gerakan lengan. Arduino mega mengontrol gerakan lengan dengan cara membaca data dari receiver kemudian diolah dengan inverse kinematic.

Gambar 2.16 adalah diagram alir keseluruhan program pada Arduino Mega. Arduino Mega memiliki empat input dan empat output. Input yang pertama pada Arduino Mega adalah pemilihan mode. Pemilihan mode pada Arduino Mega dilakukan menggunakan remot. Pemilihan mode dilakukan dengan saklar C pada remot. Setiap perintah yang dikirim dari remot diterima oleh receiver. Arduino Mega membaca sinyal PWM dari receiver.

Pada Arduino, pembacaaan sinyal PWM menggunakan fungsi yang telah disediakan oleh Arduino, yaitu *void pulseIn(void)* . PWM yang dibaca oleh Arduino memiliki kisaran 1300 sampai dengan 2400 microsecond. Dengan memanfaatkan kisaran tersebut dibuatlah suatu kondisi pemilihan mode ataupun perintah yang lain. Pemilihan mode menghasilkan menghasilkan output data yang dikirim ke Arduino Uno.

Data yang dikirim dari Arduino Mega menuju Arduino Uno adalah data dua bit.



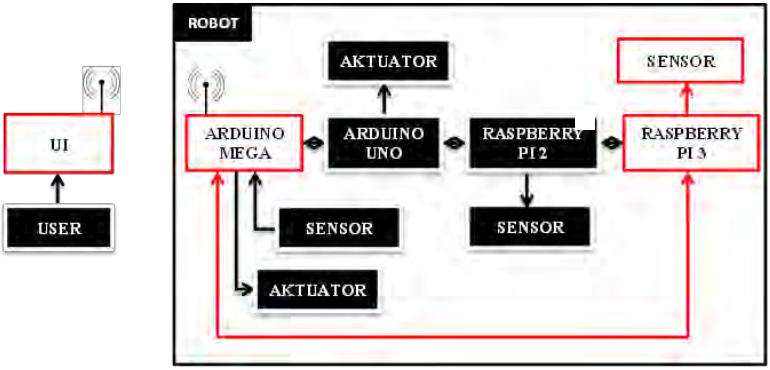

**Gambar 2.20** Diagram alir programsistem awal pada Arduino Mega[1]

# BAB III

# PERANCANGAN SISTEM

Pada bab ini akan dibahas perancangan keseluruhan sistem dimulai dari penjelasan sistem secara global hingga merujuk pada bagian-bagian spesifik. Perancangan sistem pada tugas akhir ini lebih ditekankan pada pengembangan metode pengambilan objek berbahaya, sesuai dengan judul tugas akhir ini. Pengembangan yang dilakukan akan dibahas pada bab ini, sehingga pengembangan dapat dipahami.

## 3.1 Diagram Blok Sistem




**Gambar 3.1** Diagram Sistem Robot

Pada gambar 3.1 dapat dilihat diagram blok sistem yang digunakan pada tugas akhir ini. Kotak dengan blok hitam dimaksudkan untuk memberikan gambaran bahwa tidak terdapat pengembangan. Sedangkan blok dengan warna putih dimaksudkan memberikan gambaran bahwa pengembangan dilakukan pada bagian tersebut. Penjelasan mengenai sistem awal dan pengembangan yang dilakukan dapat dilihat pada bagian selanjutnya.

Pada awalnya, sistem hanya terdapat tiga unit proses yaitu Arduino Mega, Arduino Uno, dan Raspberry Pi 2. Masing-masing dari unit proses ini memiliki tugas dan fungsi masing-masing. Pada Arduino Mega terdapat sensor berupa sensor ultrasonik dan aktuator berupa servo. Pada Arduino Uno, hanya terdapat aktuator berupa

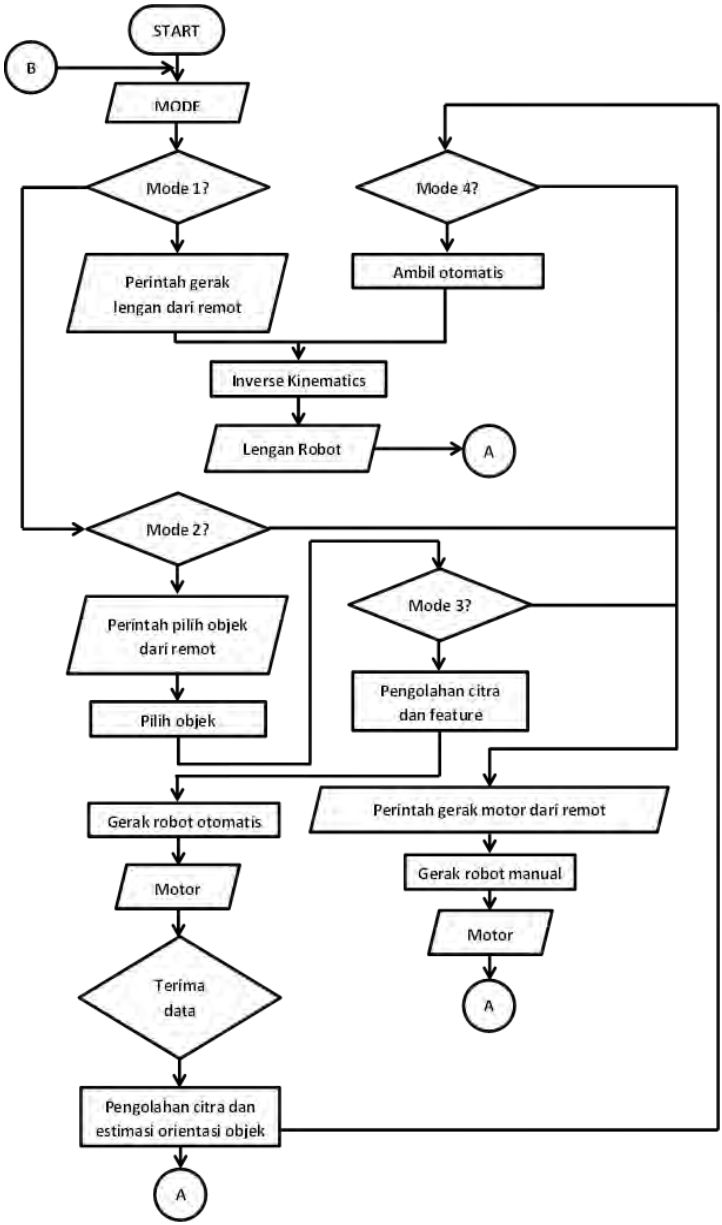
motor yang digunakan untuk menggerakkan motor dari posisi A ke posisi B. Sedangkan Raspberry Pi 2 digunakan sebagai unit proses pengolah citra untuk *tracking object.*

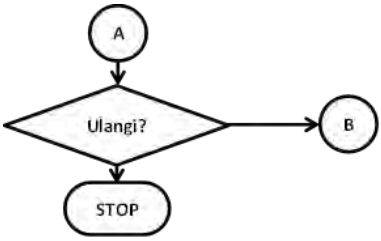
## 3.2 Pengembangan Sistem

Bagian dari diagram blok sistem pada tugas akhir ini dapat dikelompokkan menjadi *user*/pengguna, *user interface*, Arduino Mega, Arduino Uno, Raspberry Pi 2, dan Raspberry Pi 3. Antara *user interface* yang digunakan dengan arduino mega terdapat transmitter dan receiver yang digunakan sebagai penghubung kedua jenis komponen pada tugas akhir ini. Bentuk dari *user interface* sendiri cukup beragam, tetapi yang digunakan pada tugas akhir ini adalah *user interface* berupa radio kontrol dan lcd monitor.

Pada penelitian ini digunakan empat mode yang dapat diinputkan melalui radio kontrol. Mode pertama ialah mode manual. Pada mode ini, pengguna dapat mengontrol robot secara manual. Manual yang dimaksudkan adalah manual secara keseluruhan, semisal manual menjalankan robot dari posisi A ke posisi B, manual menggerakkan lengan, ataupun manual menggerakkan gripper. Mode kedua, ialah mode memilih objek. Pada mode ini dapat dipilih objek yang ingin kita ambil, tentu dengan bantuan layar lcd monitor sebagai respon visual terhadap pengguna. Mode ketiga adalah mode mendekati objek. Mode ini dapat dijalankan setelah objek yang akan diambil telah dipilih. Robot akan mendekati objek dengan bantuan pengolahan citra dan sensor ultrasonik guna mengoptimalisasi *tracking object* dari objek yang telah dipilih. Setelah robot mendeteksi objek berada pada jarak yang telah ditentukan, yaitu 12 cm, robot akan berhenti dan melakukan estimasi pose. Mode keempat ialah mode mendekati dan mengambil objek. Hampir sama dengan mode ketiga, hanya saja mode keempat ini memiliki kemampuan untuk langsung mengambil objek yang telah dipilih. Pengambilan dilakukan setelah estimasi orientasi objek dilakukan.

Pengembangan sistem yang dilakukan, difokuskan pada metode pengambilan. Hal ini dimaksudkan agar robot dapat lebih aplikatif dalam penggunaannya. Kemampuan robot untuk mengambil suatu objek dalam orientasi apapun secara otomatis membuat robot lebih aplikatif dalam penggunaan dan lebih mudah dikendalikan. Maksud dari mudah dikendalikan adalah pengguna dapat dengan mudah memahami cara penggunaan robot tersebut.


START
B
MODE
Mode 1?
Mode 4?
Perintah gerak lengan dari remot
Ambil otomatis
Inverse Kinematics
Lengan Robot
A
Mode 2?
Mode 3?
Perintah pilih objek dari remot
Pengolahan citra dan feature
Pilih objek
Gerak robot otomatis
Perintah gerak motor dari remot
Motor
Gerak robot manual
Motor
Terima data
A
Pengolahan citra dan estimasi orientasi objek
A



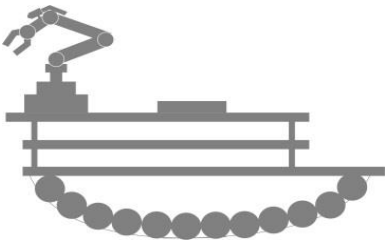

**Gambar 3.2** Diagram alir keseluruhan program

Secara umum cara kerja dari sistem pada penelitian kali ini dapat dirangkum dalam satu paragraf. Pada penelitian ini *user/*pengguna adalah manusia yang mengoperasikan robot atau sistem. Sedangkan *user interface* adalah suatu antarmuka yang memiliki fungsi agar manusia dan mesin dapat berkomunikasi dengan baik. Jadi, langkah pertama ialah pengguna mengatur radio kontrol pada kondisi mode manual. Langkah pertama inilah yang digunakan pengguna untuk menemukan letak objek berbahaya. Dibantu dengan gambar dari kamera navigasi yang ditampilkan melalui lcd monitor, pengguna dapat mencari objek secara manual. Setelah objek yang akan diambil ditemukan, pengguna dapat mengatur posisi robot untuk mendekati objek. Setelah posisi robot dirasa sudah cukup untuk melakukan *tracking object,* dapat dilakukan pemilihan objek dengan pengaktifan mode kedua. Pengaktifan mode kedua memanfaatkan raspberry pi 2 untuk melakukan proses pengolahan citra. Setelah objek dipilih, dapat diaktifkan mode ketiga atau mode keempat yang masing-masing memiliki fungsi agar robot mampu mendekati objek dan mengestimasi orientasi objek ataupun ditambah dengan pengambilan otomatis.

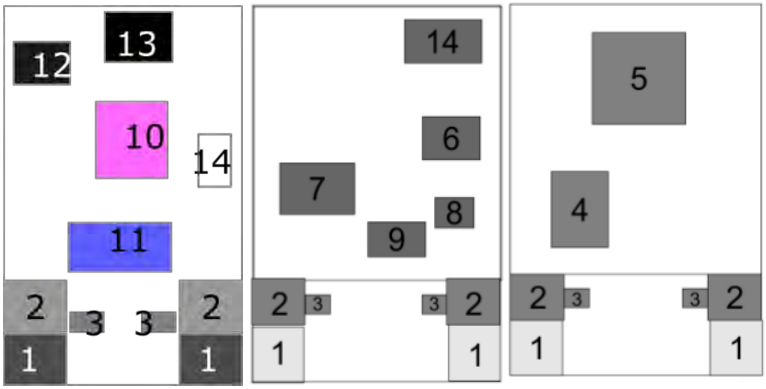
Radio kontrol digunakan sebagai antarmuka pengguna untuk menavigasikan robot. Arduino Mega digunakan sebagai pengakuisisi sensor ultrasonik dan pengatur gerakan lengan sebagai aktuator. Sensor ultrasonik digunakan pada mode ketiga dan keempat, sehingga robot akan berhenti sebelum terjadi tabrakan antara robot dengan objek. Arduino Uno digunakan untuk mengontrol laju kecepatan robot. Arduino Uno juga digunakan untuk berkomunikasi dengan Raspberry Pi 2. Data yang dikomunikasikan antara Raspberry Pi 2 menuju Arduino Uno adalah perintah untuk menggerakkan robot oleh Raspberry Pi 2. Sedangkan perintah yang dikirim dari arduino Uno

menuju Raspberry Pi 2 adalah perintah mengendalikan pointer Raspberry Pi 2 melalui remot.

Raspberry Pi 2 digunakan untuk pengolahan citra. Raspberry Pi 2 melakukan pengolahan citra dengan menggunakan *lucas kanade* pada OpenCV. Raspberry Pi 2 berkomunikasi serial dengan Arduino Uno. ESC adalah Electric Speed Control. ESC digunakan untuk menggerakkan motor DC. Pada ESC terdapat regulator arus dan tegangan sehingga dapat menatur penggunaan daya dan kecepatan motor DC. Pengontrolan navigasi dilakukan dengan prinsip *differential steer*.



**Gambar 3.3** Ilustrasi robot tampak samping




(a) (b) (c)

**Gambar 3.4** Ilustrasi robot. (a) tampak dari layer 1 (b) tampak dari layer 2 (c) tampak dari layer 3

Keterangan ilustrasi robot:

1. ESC
2. Rotary encoder
3. Motor DC
4. Raspberry Pi 3
5. Lengan Robot dan kamera
6. Video sender
7. Raspberry Pi 2
8. Multiplexer analog
9. Regulator Baterai
10. Arduino Uno
11. Arduino Mega
12. 3DR telemetry
13. Kamera
14. GPS dan Compass

Desain robot pada tugas akhir ini dapat dilihat pada gambar 3.3 dan 3.4. *Mobile* robot dirancang memiliki bentuk dasar menyerupai tank. Robot digerakkan menggunakan dua motor dc secara *differential steer*. Robot di desain berlapis untuk mempermudah peletakan komponen. Layer 1 adalah lapisan robot yang paling bawah. Kamera navigasi diletakkan pada bagian bawah layer 2. Hal ini dimaksudkan agar lingkup pandang kamera cukup luas. Kamera untuk mendeteksi objek diletakkan pada bagian atas layer 2. Lengan robot ditempatkan pada layer 3 dan pada posisi robot yang paling depan, sehingga mempermudah pergerakan lengan. Pada layer ketiga inilah pengembangan dilakukan, kamera untuk mengestimasi pose objek diletakkan pada lengan robot yang ada pada layer 3. Sehingga, lingkup pandang kamera terhadap benda didapatkan secara vertikal. Raspberry Pi 3 pun diletakkan di bagian bawah layer 3 sebagai unit proses pengolah citra untuk estimasi pose objek.

### 3.2.1 Antarmuka

Sama dengan penjelasannya sebelumnya, *user*/pengguna yang dimaksud pada bagian ini ialah manusia yang mengoperasikan robot (operator) dengan antarmuka yang tersedia. Antarmuka atau *user interface* yang digunakan adalah radio kontrol tipe Radiolink AT9 dan 7'' LCD Monitor. Radio kontrol berfungsi sebagai pengontrol pergerakan robot, sedangkan LCD Monitor difungsikan sebagai visualisasi gambar yang telah ditangkap oleh kamera analog yang terdapat di sistem. Dengan kata lain, LCD Monitor digunakan sebagai respon visual terhadap pengguna. Respon visual bagi pengguna merupakan faktor penting dalam suatu sistem. Hal ini karena respon visual bagi manusia lebih baik daripada tanpa respon visual.

Sinyal yang dikirimkan dari radio kontrol diterima oleh receiver yang terhubung langsung ke Arduino Mega. Sedangkan untuk penampilan gambar yang ditangkap dilakukan dengan cara yang sedikit berbeda. Gambar dari kamera dikirim menggunakan *video sender* dan diterima oleh *video receiver*. *Video sender* adalah perangkat untuk mengirimkan data video analog. *Video receiver* adalah perangkat untuk menerima data video analog. Data analog video tersebut kemudian ditampilkan di monitor.

Selain untuk mengontrol pergerakan robot, radio kontrol digunakan sebagai pengatur mode pada robot. Seperti dijelaskan sebelumnya, terdapat empat mode pada robot ini. Sedangkan pada lcd monitor, dapat menampilkan satu hasil visualisasi dari dua kamera yang bekerja bergantian. Kamera pertama adalah kamera analog yang digunakan untuk navigasi robot. Sedangkan kamera kedua merupakan kamera digital yang digunakan untuk menangkap gambar yang akan diolah citranya. Kamera kedua ini terhubung langsung dengan raspberry pi 2. Dengan kata lain, kamera kedua digunakan untuk *tracking object*. Pemilihan gambar yang akan ditampilkan memanfaatkan fungsi dari mode yang diinputkan dan multiplexer. Jika mode manual yang diinputkan, multiplexer akan memilih gambar dari kamera analog yang akan ditampilkan. Jika mode lain yang dipilih, maka hasil gambar dari kamera kedua lah yang akan divisualisasikan oleh lcd monitor.

**Gambar 3.5** Radio Kontrol sebagai antarmuka


**Gambar 3.6** *Radio receiver, Video sender* dan *Video receiver*

**Gambar 3.7** Hasil LCD Monitor

**Gambar 3.8** Mode Robot Pada Radio Kontrol

Radio kontrol juga digunakan untuk menggerakkan kamera analog yang berfungsi sebagai navigator. Kamera dapat digerakkan menggunakan motor servo, yaitu sebagai output dari proses menggerakkan kamera. Gerakan kamera ini menggunakan tuas kiri pada kamera. Ilustrasi gerakan motor servo kamera dan input dari remot dapat dilihat pada gambar di bawah ini.


**Gambar 3.9** Ilustrasi gerakan kamera (kanan) diiringi dengan input dari remot (kiri)

Gerakan memutar capit dan mencapit secara manual ataupun otomatis dapat direalisasikan dengan radio kontrol. Gerakan memutar capit sangat penting karena diperlukan untuk menyesuaikan capit terhadap orientasi benda. Capit dapat diputar dengan pertambahan sekitar $1^{o}$. Kemudian gerakan mencapit diperlukan untuk menyesuaikan capit dengan benda. Pada gerakan mencapit ini hanya digunakan dua gerakan, yaitu membuka dan menutup.

### 3.2.2 Arduino Uno

Pada Tugas akhir ini Arduino Uno digunakan untuk mengontrol laju kecepatan robot seperti pada penelitian sebelumnya. Pengendalian laju kecepatan robot tidak dibuat satu sistem dengan Arduino Mega karena dapat mengganggu performa Arduino Mega. Spesifikasi Arduino Uno yang digunakan adalah sebagai berikut:

| | |
|---|---|
| Mikrokontroler | ATmega328P |
| Tegangan operasi | 5V |
| Tegangan input | 7-12V |
| Input Voltage (limit) | 6-20V |
| Pin I/O | 14 (6 diantara menghasilkan dapat PWM) |
| Arus DC pada pin I/O | 20 mA |

| | |
|---|---|
| Flash Memory | 32 KB, 0.5 KB digunakan sebagai bootloader |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Frekuensi Clock | 16 MHz |


**Gambar 3.10** Arduino Uno

**Tabel 3.1** Data komunikasi Arduino Uno ke Raspberry Pi 2

| **Perintah** | **Kode** | **Pin 11** | **Pin 10** | **Pin 9** |
|---|---|---|---|---|
| Perintah *click* | Mode 3 | HIGH | LOW | HIGH |
| Perintah *clear* | Mode 1 | HIGH | HIGH | LOW |
| Navigasi kursor keatas | Mode 2 | LOW | HIGH | HIGH |
| Navigasi kursor kebawah | | LOW | LOW | LOW |
| Navigasi kursor kekanan | | LOW | LOW | HIGH |
| Navigasi kursor kekiri | | LOW | HIGH | LOW |
| Navigasi diam | | HIGH | LOW | LOW |

Diagram alir program arduino pada tugas akhir ini dapat dilihat pada gambar 2.11. Sesuai dengan diagram alir pada penelitian sebelumnya, Arduino Uno juga difungsikan sama seperti sebelumnya. Arduino Uno memiliki empat input. Dua input pertama dapat dilihat pada diagram alir, sedangkan dua input yang kedua terdapat dalam

proses “Mode navigasi robot mendekati objek”. Saat Arduino Uno pertama kali dijalankan, Arduino Uno lebih dahulu mengenali input mode dari Arduino Mega.

Mode pertama adalah mode untuk menggerakkan robot secara manual. Dalam proses penerjemahan radio kontrol, perintah dari radio kontrol tidak langsung menuju Arduino, namun melalui receiver dan Ardupilot. Receiver menerjemahkan perintah dari radio kontrol menjadi sinyal PWM, kemudian sinyal PWM diteruskan ke Ardupilot. Didalam proses Ardupilot terdapat penerjemahan perintah radio kontrol menjadi perintah pengendalian robot dengan dua motor. Namun, output perintah dari Ardupilot dibaca oleh Arduino Uno untuk diteruskan ke motor. Hal ini dilakukan supaya Arduino Uno dapat memegang kendali motor robot.

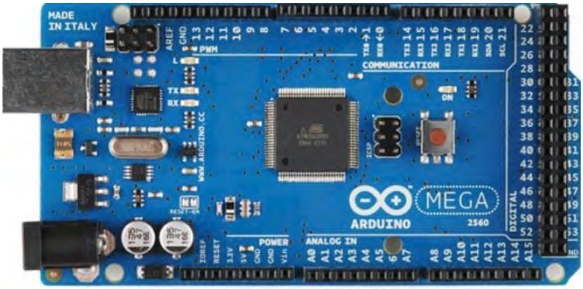
Mode kedua adalah mode penerjemahan radio kontrol menjadi perintah menavigasikan kursor. Sama seperti pada mode pertama, perintah dari radio kontrol di terjemahkan tidak langsung dari radio kontrol, namun melalui receiver dan Ardupilot. Pada proses ini Arduino Uno menerjemahkan perintah dari Ardupilot menjadi perintah menggerakkan kursor. Perintah menggerakkan kursor dikirim dari Arduino Uno ke Raspberry Pi menggunakan pin GPIO pada Arduino Uno dan Raspberry Pi 2. Data navigasi tersebut digabung dengan data perintah *click* dan *clear,* isi data tersebut dapat dilihat pada tabel 1.

Mode ketiga adalah mode otomatis robot mendekati objek. Didalam mode ini terdapat proses navigasi robot mendekati objek dan kontrol jarak. Dalam mode ini terdapat output perintah *click* menuju Raspberry Pi 2 dan output motor robot. Sehingga hasil dari mode ini robot akan bergerak otomatis mendekati objek. Pada mode keempat, hal yang sama terjadi hanya dengan perbedaan penambahan fungsi pengambilan objek secara otomatis.

### 3.2.3 Raspberry Pi 2

Raspberry Pi 2 digunakan untuk pengolahan citra. Raspberry Pi 2 melakukan pengolahan citra dengan menggunakan *lucas kanade* pada OpenCV. Dengan kata lain, Raspberry Pi 2 pada tugas akhir ini tidak mengalami pengembangan apapun atau fungsinya sama seperti penelitian sebelumnya. Diagram alir program dari Raspberry Pi 2 dapat dilihat pada gambar 2.13. Contoh gambar kursor yang dihasilkan pun dapat dilihat pada gambar 2.15

### 3.2.4 Arduino Mega



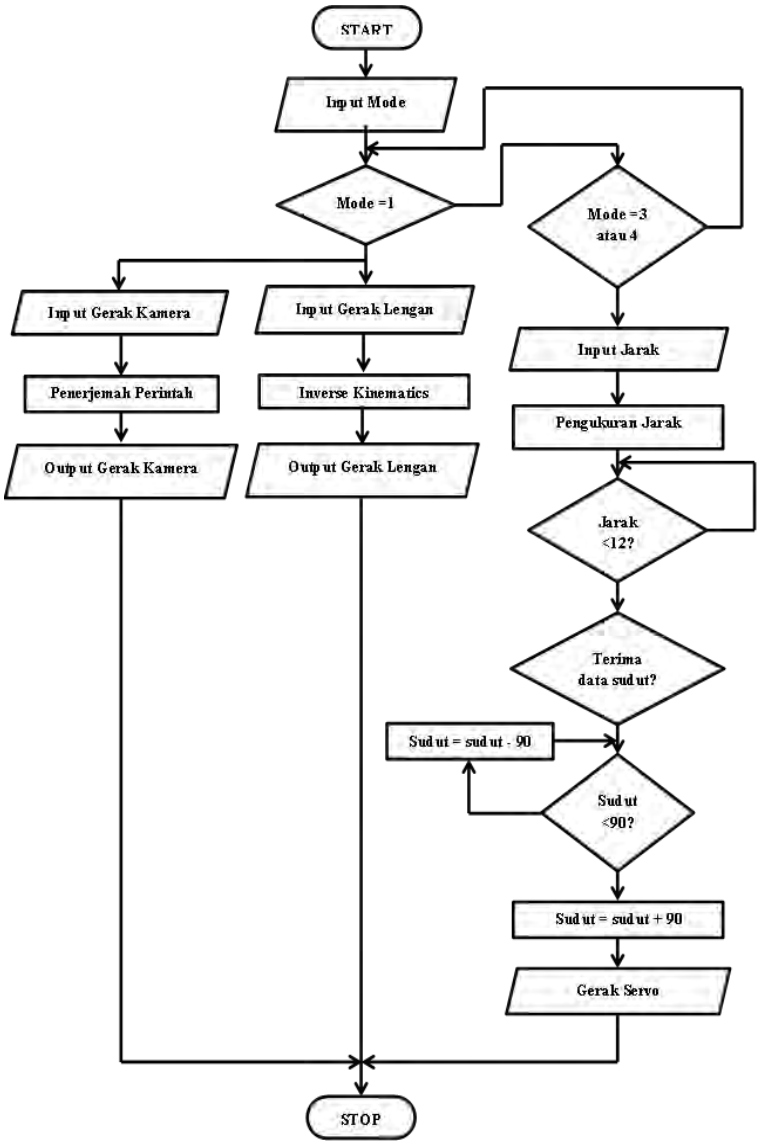

**Gambar 3.11** Arduino Mega

Berikut ini adalah spesifikasi singkat Arduino Mega [2]:

| | |
|---|---|
| Microcontroller | ATmega1280 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limits) | 6-20V |
| Digital I/O Pins | 54 (15 diantaranya menghasilkan output PWM) |
| Analog Input Pins | 16 |
| DC Current per I/O Pin | 40 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 128 KB (4 KB dari memori digunakan sebagai bootloader) |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Clock Speed | 16 MHz |

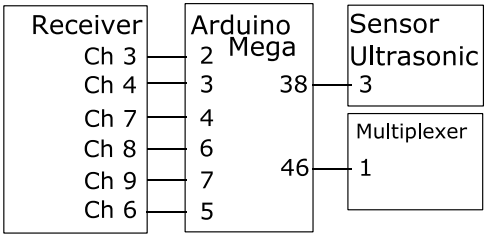
Diagram alir Arduino Mega pada tugas akhir ini mengalami pengembangan dari penelitian sebelumnya. Pada tugas akhir sebelumnya, Arduino Mega hanya mendeteksi ada tidaknya objek di depan robot. Pada tugas akhir ini, robot diharapkan mampu memberikan respon pada orientasi objek. Sehingga, Arduino Mega memiliki andil besar dalam hal ini, karena Arduino Mega akan memberikan efek gerak pada ujung lengan.

Dapat dikatakan bahwa tiap proses yang terjadi selalu melalui Arduino Mega terlebih dahulu. Hal ini membuat Arduino Mega layaknya otak manusia. Sehingga seluruh pergerakan robot dikontrol oleh Arduino Mega. Meskipun pergerakan aktuator tidak selalu berhubungan dengan Arduino Mega secara langsung.

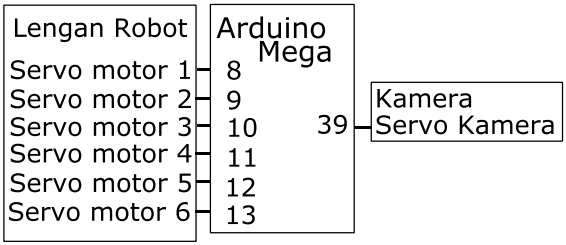



**Gambar 3.12** Diagram alir Arduino Mega

Jika dianalogikan sebagai organ manusia, maka Arduino Mega pada tugas akhir ini berfungsi seperti otak manusia. Hal ini karena Arduino Mega lah yang menjadi pusat kontrol dan informasi dari robot. Arduino Mega digunakan untuk mengontrol gerakan lengan robot. Pada Arduino Mega terdapat program *inverse kinematics* lengan, sehingga dapat memudahkan pengguna dalam mengontrol gerakan lengan. Arduino Mega mengontrol gerakan lengan dengan cara membaca data dari receiver kemudian mengolahnya dengan *inverse kinematics*. Pada tugas akhir ini, Arduino Mega juga digunakan dalam komunikasi dengan Raspberry Pi 3. Raspberry Pi 3 tersambung dengan kamera, sehingga terdapat pengolahan citra pada unit proses ini. Data pengolah citra inilah yang nantinya akan dikirimkan oleh Raspberry Pi 3 ke Arduino Mega, sehingga motor servo pada ujung lengan akan berputar sesuai perhitungan dari pose objek yang terdeteksi oleh pengolah citra.

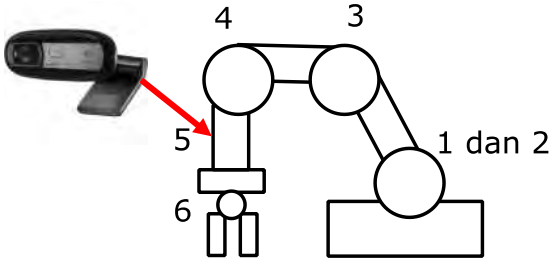



**Gambar 3.13** Pengkabelan Arduino Mega dengan receiver, multiplexer dan sensor ultrasonik

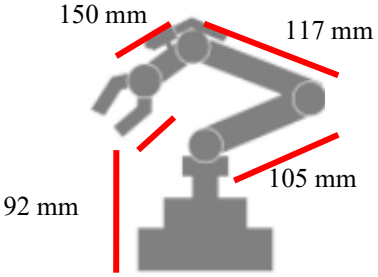



**Gambar 3.14** Pengkabelan Arduino Mega dengan motor servo pada kamera analog dan lengan robot

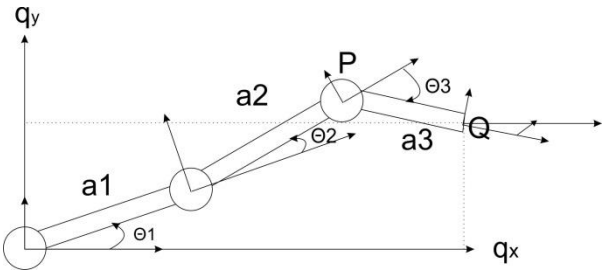
Lengan robot terdiri dari 6 Dof. Sambungan (joint) lengan robot terbuat dari motor servo TowerPro MG996R. Lengan robot terbuat dari bracket standard servo. Lengan robot terdiri dari enam motor servo dilengkapi dengan satu kamera di ujungnya. Motor servo pertama dan kedua adalah dua buah motor servo yang dirangkap. Motor servo pertama dan kedua dirangkap untuk menambah daya motor servo sehingga motor servo dapat mengangkat beban yang lebih besar. Motor servo yang ketiga dan keempat adalah motor servo biasa yang terangkai secara seri. Gerakan motor servo pertama, kedua, ketiga, dan keempat diatur menggunakan *inverse kinematics*. Motor servo yang kelima berfungsi sebagai pemutar orientasi capit. Orientasi capit pada mode manual dapat diputar oleh pengguna sehingga pengguna dapat menyesuaikan orientasi capit terhadap orientasi benda. Pada mode ketiga, yaitu mode mendekati objek, motor servo kelima digunakan sebagai aktuator saat nilai orientasi objek telah didapatkan dari hasil pengolahan citra. Motor servo keenam adalah motor servo untuk menggerakkan capit. Capit pada motor servo hanya digerakkan dengan mode membuka dan menutup. Kamera ini digunakan untuk mendeteksi orientasi objek yang akan diambil.




**Gambar 3.15** Peletakan enam motor servo dan kamera




**Gambar 3.16** Rincian ukuran lengan robot




**Gambar 3.17** Lengan 4 DoF

$$p_x{}^2 + p_y{}^2 = a_1{}^2 + a_2{}^2 + 2a_1a_2\cos\ _2 \tag{18}$$

$$_2 = \cos^{-1}\left(\frac{p_x{}^2 + p_y{}^2 - a_1{}^2 - a_2{}^2}{2a_1a_2}\right) \tag{19}$$

$$_1 = \tan^{-1}\left(\frac{p_y}{p_x}\right) - \tan^{-1}\left(\frac{a_2\sin\ _2}{a_1 + a_2\cos\ _2}\right) \tag{20}$$

$$\theta_3 = \phi -\ _1 - \theta_2 \tag{21}$$

(a)

(b)

**Gambar 3.18** (a) Ilustrasi putaran capit (b) Ilustrasi capit menutup dan capit membuka

Arduino Mega memiliki fungsi dalam menggerakkan lengan yang diproses menggunakan *inverse kinematics*. Pada tugas akhir ini, digunakan lengan robot 4 Dof yang sama dengan penelitian sebelumnya. Ilustrasi gambar lengan 4 Dof dengan *inverse kinematics* yang digunakan pada tugas akhir ini dapat dilihat di atas.

Hubungan antara Arduino Mega dengan sensor ultrasonik pada tugas akhir ini ialah Arduino Mega menjadi unit proses untuk mengetahui jarak yang didapatkan dari pengukur jarak. Jarak tersebut yang digunakan sebagai peringatan bahwa benda sudah berada pada jarak tertentu. Output peringatan diumpankan pada unit proses Arduino Uno. Komunikasi yang digunakan untuk menghantarkan data ini hanyalah berupa data satu bit sehingga komunikasi ini hanya menggunakan GPIO. Berikut ini adalah tabel data yang dikirim.

**Tabel 3.2** Pengiriman data peringatan jarak ke Arduino Uno

| **Input Jarak** | **Kode** | **Pin 43** |
|---|---|---|
| Jarak kurang dari 12 cm | Ping_ok = 0 | LOW |
| Jarak lebih dari 12 cm | Ping_ok = 1 | HIGH |

Input lain yang terdapat di Arduino Mega merupakan pengembangan sistem pada Tugas Akhir ini. Input tersebut berasal dari komunikasi serial antara Arduino Mega dengan Raspberry Pi 3. Raspberry Pi 3 yang digunakan sebagai pengolah citra mengirimkan data orientasi objek. Data tersebut kemudian diolah di Arduino Mega dan dioutputkan dalam bentuk putaran servo pada ujung lengan. Sehingga, lengan mampu menyesuaikan orientasi objek.

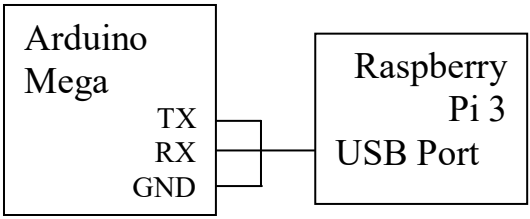
Data yang diterima oleh Arduino Mega dari Raspberry Pi 3 merupakan data sudut. Data ini kemudian diolah menurut algoritma berikut.

1. Apabila data sudut yang diterima lebih dari 90. Maka nilai sudut dikurangkan dengan 90.
2. Apabila data sudut yang diterima kurang dari 90. Maka nilai ditambahkan dengan 90.

Hasil komputasi inilah yang kemudian menjadi output dari Arduino Mega. Output ini dikeluarkan dalam bentuk putaran motor servo, yaitu motor servo pada ujung lengan. Sehingga, lengan dapat secara autonomous mengestimasi orientasi objek yang terdeteksi.

Gambar 3.20 di bawah merupakan ilustrasi komunikasi serial antara Arduino Mega dengan Raspberry Pi 3. Komunikasi serial Arduino Mega dengan Raspberry Pi 3 dilakukan dengan bantuan *USB*

*to TTL*. Lewat *USB to TTL* inilah Arduino Mega mampu mengirimkan data ke Raspberry Pi 3, begitu juga sebaliknya, Raspberry Pi 3 mampu mengirimkan data ke Arduino Mega melalui komunikasi serial tersebut. Arduino Mega mengirimkan data "aa" ke Raspberry Pi 3 saat robot sudah berhenti atau mendeteksi adanya benda di depan robot. Apabila Raspberry Pi 3 telah menerima data tersebut, kamera akan mulai mengambil gambar sehingga proses pengolahan citra oleh Raspberry Pi 3 pun dilaksanakan.




**Gambar 3.19** Komunikasi Serial Arduino Mega dan Raspberry Pi 3

### 3.2.5 Raspberry Pi 3

Raspberry Pi 3 digunakan untuk pengolahan citra. Raspberry Pi 3 melakukan pengolahan citra dengan menggunakan *hough line transform* pada OpenCV. OpenCV adalah library pemrograman yang berisi berbagai fungsi pengolahan citra. Sedangkan *hough line transform* adalah suatu metode yang digunakan untuk mendeteksi adanya garis dengan fitur-fiturnya.

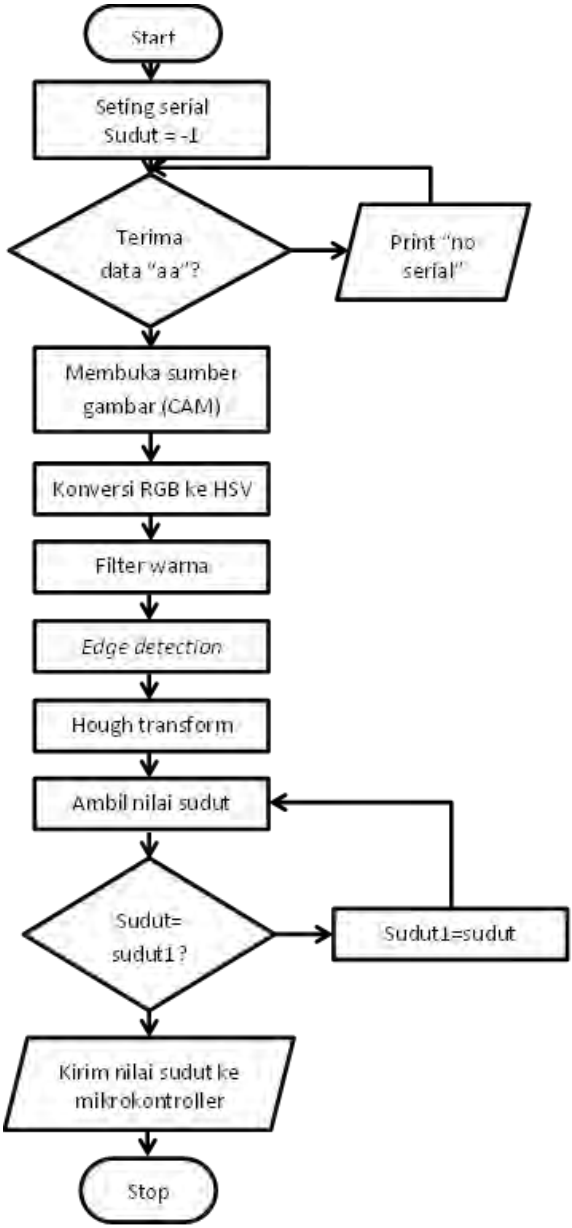
Raspberry Pi 3 berkomunikasi serial dengan Arduino Mega. Komunikasi dari Arduino Mega berisi perintah untuk mengaktifkan pengolahan citra dari kamera untuk mengetahui orientasi objek. Data orientasi yang didapat pada proses ini dikirimkan kembali secara serial ke Arduino Mega untuk diolah dalam Arduino Mega. Setelah diolah, diharapkan servo pada ujung lengan akan bergerak sesuai hasil komputasi pada Arduino Mega. Berikut adalah spesifikasi dari Raspberry Pi 3:

| | |
|---|---|
| **SoC** | Broadcom BCM2837 |
| **CPU** | 4× ARM Cortex-A53, 1.2GHz |
| **GPU** | Broadcom VideoCore IV |
| **RAM** | 1GB LPDDR2 (900 MHz) |
| **Networking** | 10/100 Ethernet, 2.4GHz 802.11n wireless |

| | |
|---|---|
| **Bluetooth** | Bluetooth 4.1 Classic, Bluetooth Low Energy |
| **Storage** | microSD |
| **GPIO** | 40-pin header, populated |
| **Ports** | HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface, Display Serial Interface |

Kamera yang digunakan pada estimasi orientasi objek adalah kamera Logitech C170. Kamera yang digunakan pada *object tracking* juga berjenis Logitech C170. Kamera untuk mendeteksi orientasi objek ini diletakkan pada lengan robot, seperti pada gambar 3.17. Kamera ini memiliki spesifikasi sebagai berikut:

| | |
|---|---|
| **Kebutuhan Sistem** | 1 GHz (Direkomendasikan 1.6 GHz)<br>RAM 512 MB atau lebih<br>Kapasitas hard drive 200 MB<br>Koneksi Internet<br>Port USB 1.1 (direkomendasikan 2.0) |
| **Spesifikasi Teknikal** | Panggilan video (640 x 480 piksel) dengan sistem yang direkomendasikan<br>Merekam video: Hingga 1024 x 768 piksel<br>Teknologi Logitech Fluid Crystal™*<br>Foto: Hingga 5 megapiksel (ditingkatkan menggunakan perangkat lunak)<br>Mikrofon bawaan dengan pengurangan noise<br>Bersertifikat Hi-Speed USB 2.0 (direkomendasikan)<br>Klip universal cocok dengan berbagai laptop, monitor LCD atau CRT |
| **Perangkat lunak** | Kontrol pan, miring, dan zoom<br>Merekam video dan memotret foto<br>Penelusuran wajah<br>Pendeteksian gerakan |




**Gambar 3.20** Diagram alir program pada Raspberry Pi 3

Tidak seperti dua kamera yang lain, gambar hasil tangkapan dari kamera ini tidak dapat dilihat via layar. Karena, hal ini dirasa akan mengganggu kinerja robot. Fungsi *Houghline transform* yang dipakai cukup memakan RAM sehingga jika gambar ditampilkan, tentu kinerja Raspberry Pi 3 akan mempengaruhi kinerja robot.

Berbeda dengan penelitian sebelumnya yang hanya memakai tiga unit proses, pada Tugas Akhir ini ditambahkan satu unit proses yaitu Raspberry Pi 3. Hampir sama dengan Raspberry Pi 2, Raspberry Pi 3 digunakan sebagai pengolah citra. Namun, kegunaan dari dua unit proses ini berbeda. Apabila Raspberry Pi 2 digunakan sebagai *tracking object*, Raspberry Pi 3 digunakan sebagai pengolah citra dalam pengestimasi orientasi objek.

Perancangan sistem autonomous diberikan dengan penambahan kamera. Estimasi orientasi objek yang diletakkan pada lengan robot dilakukan dengan pemberian program vision dengan OpenCV. Kamera pertama kali menangkap gambar dengan format RGB, lalu dikonversi ke dalam format HSV. Dari gambar HSV ini dilakukan filter warna. Untuk melakukan ini, diperlukan pengaturan nilai *hue* pada HSV. Warna yang akan diambil pada program ini adalah warna merah, sehingga nilai H atau *hue* yang diambil berada pada range 0 sampai 10 dan 160 sampai 179. Setelah didapat objek yang diinginkan dengan warna merah, dilakukan *edge detection.* Jenis *edge detection* yang digunakan pada Tugas Akhir ini adalah *Canny Edge Detection.* Setelah *edge detection* diterapkan, dilakukan deteksi garis dengan *hough transform*. Dari *hough transform* didapatkan nilai sudut objek berbahaya.

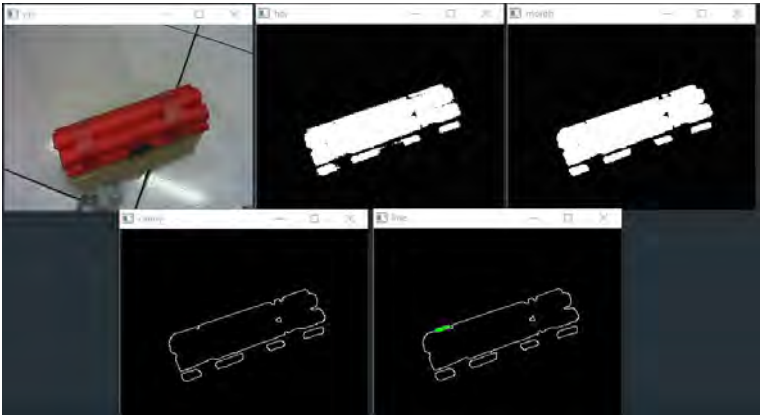
Secara mendetail metode yang digunakan untuk mengestimasi pose objek adalah:

1. Mengonversi gambar RGB yang ditangkap ke dalam ruang warna HSV. Dalam ruang warna HSV, nilai warna merah, kuning, hijau dan lain-lain jauh lebih presisi daripada dalam ruang RGB. Pengkonversian gambar RGB ke HSV dilakukan dengan memanfaatkan rumus perubahan RGB ke HSV seperti pada persamaan (4). Di dalam opencv juga sudah terdapat fungsi untuk mengkonversi gambar RGB ke HSV, sehingga pengkonversian dapat dilakukan dengan lebih sederhana. Pada penelitian ini, digunakan fungsi cv2.COLOR_BGR2HSV. Fungsi ini mampu mengonversi ruang warna RGB ke HSV.

2. Filter warna. Dari ruang warna HSV, difilter/disaring satu warna yaitu merah. Pemilihan warna merah karena salah satu objek berbahaya berwarna merah, yaitu dinamit. Untuk filter warna merah pada gambar HSV digunakan segmen *hue* pada HSV. *Hue* warna merah memiliki nilai 0 sampai 10 dan 160 sampai 179 dengan nilai S dan V berkisar pada 100 sampai 255. Pada opencv terdapat fungsi untuk mendefinisikan warna yang akan di filter dari gambar HSV. Pertama, dengan fungsi cv2.inRange diambil dua range nilai hue untuk warna merah yang sudah disebutkan. Fungsi ini sekaligus memfilter warna merah dari gambar. Kemudian, dua hasil filter gambar tersebut disatukan dengan cv2.addWeighted.
3. Morfologi. Gambar hasil filter warna diberikan fungsi morfologi. Fungsi morfologi yang digunakan adalah morfologi open. Pada opencv terdapat fungsi sederhana untuk mengaplikasikannya, yaitu cv2.MORPH_OPEN. Morfologi digunakan untuk memberikan hasil gambar yang lebih tegas.
4. Deteksi tepi. Gambar yang telah disaring, kemudian dideteksi tepinya menggunakan *Canny edge detection.*
5. *Hough Transform*. Setelah didapatkan tepi-tepi objek pada gambar, diaplikasikan fungsi *hough transform.* Pada penelitian kali ini, digunakan *hough line transform.* Hal ini dimaksudkan untuk mendeteksi garis dari benda yang memiliki panjang (berbentuk kotak). Fungsi ini merupakan inti dari sistem agar robot mampu mengestimasikan orientasi objek. Mengenai persamaan *hough transform,* dapat dilihat pada subbab 2.1.6. Pada opencv terdapat beberapa fungsi *hough transform.* Namun, pada penelitian ini digunakan fungsi cv2.HoughLinesP. Fungsi ini dipilih karena garis yang akan dideteksi adalah garis lurus. Selain itu, fungsi ini jauh lebih bagus hasilnya jika dibandingkan dengan fungsi cv2.HoughLine. Fungsi ini diatur agar mampu mendeteksi garis dengan panjang minimal 50 mm. Hal ini dapat dilihat pada lampiran program Raspberry Pi 3.
6. Ambil data sudut. Dari fungsi *hough line transform*, dapat diperoleh nilai sudut. *Hough line transform(cv2.HoughLinesP)* yang ada pada opencv sudah memiliki unsur sudut objek yang dideteksi di dalamnya. Unsur sudut pada fungsi *hough transform* berasal dari nilai x dan y yang terdeteksi yaitu x1,x2, y1, dan y2. Dengan mengaplikasikan rumus:

$$\Theta = \tan^{-1}\left(\frac{y2-y1}{x2-x1}\right) \quad (25)$$

didapatkan data berupa sudut. Data sudut inilah yang dipakai untuk estimasi pose objek. Nilai sudut yang terdeteksi kemudian diatur, sehingga memberikan hasil nilai sudut pada range $0^o$-$180^o$. Nilai ini kemudian dikirimkan ke Arduino Mega lewat komunikasi serial.



**Gambar 3.21** Metode pengestimasian orientasi objek

Jika diilustrasikan hasil masing-masing dari tiap metode yang dilakukan, dapat dilihat pada Gambar 3.31. Pada gambar dapat dilihat gambar RGB sebagai gambar original, hasil dari konversi gambar RGB ke HSV, hasil filter warna merah, deteksi tepi, *hough line transform,* dan nilai sudut.

## 3.3 Proses Estimasi Orientasi Objek

Estimasi orientasi objek hanya terjadi jika robot memasuki mode 3 atau 4. Saat mode tersebut, robot mampu mengestimasikan orientasi objek jika sudah memenuhi suatu kondisi. Kondisi yang dibutuhkan adalah objek sudah terdeteksi berada pada jarak kurang dari 12 cm. Jika kondisi tersebut sudah terpenuhi, maka fungsi estimasi orientasi dari robot akan aktif.

Saat pengguna mengaktifkan mode tiga, robot mampu mendekati objek secara otomatis. Robot mengaktifkan sensor ultrasonik, sehingga robot dapat mengetahui jarak robot dengan objek

yang dituju. Saat jarak robot dan objek kurang dari 12 cm, robot akan berhenti. Arduino Mega pada robot akan memberikan perintah kepada Raspberry Pi 3 untuk mengaktifkan kamera pada lengan. Kamera ini difungsikan sebagai pengestimasi orientasi objek. Saat kamera aktif, robot mulai mengestimasikan orientasi objek. Kamera dan Raspberry Pi 3 akan berhenti mengestimasikan orientasi objek setelah didapatkan dua nilai estimasi orientasi yang sama dan berurutan. Saat itu pula nilai estimasi dikirimkan ke Arduino Mega yang kemudian dikalkulasi dan hasilnya dikirimkan berupa sinyal pwm ke servo *gripper*.

Sedangkan saat pengguna mengaktifkan mode empat, robot tidak hanya melakukan hal yang sama seperti pada mode tiga. Robot juga akan mengambil objek secara otomatis. Maksudnya, setelah berhasil mendekati dan mengestimasi objek, lengan robot akan bergerak menurut rumus *inverse kinematics* yang digunakan. Saat lengan sudah berada pada posisi yang dituju, robot akan mencapit objek dan mengangkatnya.

# BAB IV

# PENGUJIAN DAN ANALISIS

Bab ini menjelaskan pengujian sistem sehingga dapat diketahui kinerjanya. Pengujian dilakukan di dalam ruangan atau indoor. Pada bab ini akan dibahas pengujian estimasi orientasi objek, pengujian sistem menurut iluminansi cahaya, pengujian pergerakan griper, dan pengujian lama pengambilan objek. Pada pengujian ini pun akan diaplikasikan rumus kesalahan guna mendapatkan nilai kesalahan. Rumus tersebut ialah:

$$\text{Kesalahan} = \left(\frac{\text{Nilai sebenarnya} - \text{Nilai terdeteksi}}{\text{Nilai sebenarnya}}\right) \text{ x } 100\%$$

Berikut adalah gambar hasil perancangan robot yang digunakan dalam pengujian.

**Gambar 4.1** Hasil Perancangan Robot

## 4.1 Pengujian Estimasi Orientasi Objek

Pengujian ini dilakukan untuk mengetahui akurasi kamera dalam mengestimasikan orientasi objek. Pengujian dilakukan dengan melihat hasil pengestimasian orientasi objek pada layar terminal Raspberry Pi 3 dan membandingkannya dengan pengukuran orientasi objek secara manual menggunakan penggaris busur. Pengestimasian dilakukan dengan ketelitian 1$^{o}$, begitu pula dengan pengukuran manual.

**Gambar 4.2** Grafik pengujian estimasi orientasi objek

Dari pengujian yang telah dilakukan, didapatkan beberapa nilai estimasi yang berbeda dengan seharusnya, seperti terlihat pada gambar 4.1. Selisih nilai antara estimasi dengan nilai sebenarnya berkisar pada nilai 1 derajat ke atas dan 1 derajat ke bawah. Pada pengujian orientasi objek 0 dan 180 derajat, nilai estimasi berada pada titik seharusnya, yaitu 0 dan 180 derajat. Berikut adalah tabel nilai presentase kesalahan dari pengujian ini.

**Tabel 4.1** Presentase Kesalahan Pengujian Estimasi Orientasi Objek

Dari tabel di atas terlihat bahwa nilai kesalahan terbesar ialah -7,7%. Kesalahan tersebut terjadi pada sudut sebenarnya sebesar 13$^{o}$. Meskipun kesalahan terjadi dengan perbedaan satu derajat, namun menurut perhitungan rumus kesalahan didapatkan nilai kesalahan sebesar -7,7%.

## 4.2 Pengujian Sistem Menurut Iluminansi Cahaya

Pengujian ini dilakukan untuk mengetahui kepresisian sistem dalam mengestimasikan orientasi objek menurut nilai iluminansi cahaya yang terukur. Pengujian dilakukan di ruangan (*indoor*), dan dilakukan pada nilai iluminansi yang berbeda-beda. Nilai iluminansi diukur dengan lux meter. Pengujian ini dilakukan juga untuk mengetahui kegunaan dari konversi HSV yang telah dilakukan.

Dengan mengimplementasikan perhitungan kesalahan dengan rumus kesalahan, didapatkan presentase kesalahan seperti terlihat pada tabel di bawah.

**Tabel 4.2** Presentase Kesalahan Pengujian Estimasi Orientasi Menurut Iluminansi Cahaya

Dari data di atas, terlihat bahwa kesalahan terjadi pada sebagian besar nilai iluminansi cahaya. Namun, kesalahan yang terjadi berkisar pada selisih 1 dan 2 derajat. Secara presentase, kesalahan terbesar ada pada angka -7,69%. Pada nilai iluminansi cahaya 7,1, selisih nilai sudut sebenarnya dengan nilai sudut yang terdeteksi oleh robot adalah 2 derajat. Pada nilai iluminansi 6,5 saja, robot sudah tidak mampu mengestimasikan nilai orientasi dari objek. Dengan kata lain, semakin kecil nilai iluminansi cahaya, kemungkinan kesalahan terjadi semakin besar.

## 4.3 Pengujian Pergerakan Griper

Pengujian ini dilakukan untuk mengetahui efek dari estimasi orientasi pada pergerakan servo griper. Pengujian dilakukan dengan memanfaatkan data dari estimasi orientasi objek. Data ini kemudian diolah oleh Arduino Mega yang hasil pengolahannya digunakan sebagai nilai pergerakan sudut servo griper.

Dari gambar 4.3dapat diketahui bahwa pergerakan servo griper sesuai dengan rumus yang berlaku pada Arduino Mega. Rumus/kondisi tersebut ialah jika sudut yang terdeteksi bernilai kurang dari $90^o$, maka sudut ditambahkan $90^o$. Apabila sudut yang terdeteksi di atas $90^o$, maka sudut tersebut dikurangkan $90^o$. Dari perhitungan itulah kemudian nilai hasil digunakan sebagai nilai pergerakan servo

griper. Melalui data grafik di atas pula dapat disimpulkan bahwa presentase nilai kesalahan pada pengujian ini adalah 0%.
**Gambar 4.3** Grafik pengujian pergerakan griper

## 4.4 Pengujian Lama Pengambilan Objek

Pengujian ini dilakukan untuk mengetahui lama waktu yang dibutuhkan robot untuk mengambil objek berdasarkan jarak maupun orientasi objek. Digunakan tiga nilai jarak pada pengujian ini, yaitu 15 cm, 30 cm, dan 60 cm. Pengujian dilakukan dengan melalui tahap *tracking* objek terlebih dahulu.

**Gambar 4.4** Grafik Pengujian Lama Pengambilan Objek Pada Jarak 15 cm

**Gambar 4.5** Grafik Pengujian Lama Pengambilan Objek Pada Jarak 30 cm

**Gambar 4.6** Grafik Pengujian Lama Pengambilan Objek Pada Jarak 60 cm

Dari tiga grafik di atas, dapat diketahui bahwa puncak lama waktu pengambilan objek terjadi pada orientasi sekitar 90 derajat. Pada pengujian yang dilakukan, pergerakan lengan dengan *inverse kinematics* jauh lebih lama saat orientasi berkisar pada nilai 90 derajat dibandingkan pada orientasi yang lain. Berdasarkan jaraknya, semakin besar jarak robot dengan benda, maka waktu yang dibutuhkan untuk mendekati dan mengambil objek bertambah lama.

## 4.5 Pengujian Jarak Kamera dengan Objek

Pengujian dilakukan dengan mengukur jarak kamera yang ada pada lengan robot dengan objek menggunakan penggaris, dan

mengukur orientasi objek dengan bantuan penggaris busur. Pengukuran orientasi objek digunakan untuk mengetahui nilai orientasi/sudut objek sebenarnya. Pada pengujian ini digunakan tiga nilai orientasi, yaitu 18°, 45°, dan 165°.

**Tabel 4.3** Tabel Pengujian Jarak Kamera dengan Objek

Tabel 4.3 menunjukkan hasil estimasi orientasi berdasarkan jarak yang terukur. Tabel dengan warna kuning menunjukkan nilai sudut yang terdeteksi sesuai dengan sudut sebenarnya. Pada jarak 5,5 cm didapatkan nilai sudut terdeteksi sesuai dengan nilai sudut sebenarnya. Hasil yang didapat cukup memuaskan, berdasarkan teori nilai sudut yang terdeteksi/terukur melalui *hough transform* akan mengalami kesalahan yang semakin besar jika objek semakin jauh dari kamera. Pada jarak 29,5 cm, kamera dapat memperoleh nilai sudut yang presisi pada nilai 165°. Hal ini dapat terjadi, karena nilai estimasi orientasi memiliki range pada tiap jarak yang dideteksi. Pada jarak 5,5 cm, sistem dapat secara pasti mengestimasikan nilai orientasi objek dengan tepat. Saat jarak kamera dan objek adalah 29,5 cm, sistem memiliki nilai range orientasi yang terdeteksi. Pada pengujian ini didapatkan range nilai dari 163° sampai 168°. Pada jarak 8,5 cm saja sistem sudah memiliki range pada sudut sebenarnya 165° sebesar 164° sampai 165°.

# LAMPIRAN

**Berikut ini program pada Arduino Uno:**

```
#include <Servo.h>
#define APM_MAX_KIRI 2511
#define APM_MIN_KIRI 1354
#define APM_MAX_KANAN 2501
#define APM_MIN_KANAN 1372

#define encoderKiriPinA  2
#define encoderKananPinA  3
#define MODE_BIASA      1
#define MODE_RASPBERRY  2
#define MODE_TRACKING   3
int mode = MODE_BIASA;

volatile unsigned int encoderKiriPos = 0;
volatile unsigned int encoderKananPos = 0;
long newpositionKiri,newpositionKanan;
long oldpositionKiri = 0, oldpositionKanan=0;
unsigned long newtimeKiri, newtimeKanan;
unsigned long oldtimeKiri = 0,oldtimeKanan = 0;
long velKiri, velKanan,velSetKanan=500,velSetKiri=500;
long maxSpeed=1000;
float potKanan=0, potKiri=0;

Servo motorKiri,motorKanan;
int apmKiri, apmKanan;
boolean ping_ok, interrupt_ok;
unsigned int bacaRaspberry;

void setup(){
  pinMode(encoderKiriPinA, INPUT);
  digitalWrite(encoderKiriPinA, HIGH);
pinMode(encoderKananPinA, INPUT);
  digitalWrite(encoderKananPinA, HIGH);
  pinMode(4, INPUT);     //kiri
  digitalWrite(4, HIGH);
  pinMode(5, INPUT);     //kanan
```

```
  digitalWrite(5, HIGH);
  //------------------------servo
  motorKiri.attach(6);// servo steering
  motorKanan.attach(7);// servo accelerating
  //------------------------pin Click, Clear, Kanan, Kiri, Atas, Bawah
  pinMode(11,OUTPUT);
  digitalWrite(11,HIGH);
  pinMode(10,OUTPUT);
  digitalWrite(10,LOW);
  pinMode(9,OUTPUT);
  digitalWrite(9,LOW);
  //------------------------pin sensor PING
  pinMode(8, INPUT);
  digitalWrite(8, HIGH);
  //------------------------pin input APM
  pinMode(12,INPUT);       //APM 1, motor Kiri
  digitalWrite(12,HIGH);
  pinMode(13,INPUT);       //APM 3, motor Kanan
  digitalWrite(13,HIGH);
  //------------------------pin input MODE SELECT
  pinMode(A4, INPUT);
  digitalWrite(A4, HIGH);
  pinMode(A5, INPUT);
  digitalWrite(A5, HIGH);
  Serial.begin (9600);
}

void loop (){
  //delay(500);
  if (digitalRead(8)==LOW){
    ping_ok=0;
  }
  else if (digitalRead(8)==HIGH)ping_ok=1;

if((digitalRead(A4)==LOW)&&(digitalRead(A5)==HIGH))mode=M
ODE_BIASA;
  else
if((digitalRead(A4)==HIGH)&&(digitalRead(A5)==LOW))mode=M
ODE_RASPBERRY;
```

```
  else
if((digitalRead(A4)==HIGH)&&(digitalRead(A5)==HIGH))mode=M
ODE_TRACKING;

  if (mode==MODE_BIASA){
    digitalWrite(11,HIGH);  //clear
    digitalWrite(10,HIGH);
    digitalWrite(9,LOW);
    interruptMati();
    apmKiri=avg(pulseIn (12, HIGH, 35000),3); //Pin 12, APM 1,
motor Kiri
    apmKanan=avg(pulseIn (13, HIGH, 35000),3); //Pin 13, APM 3,
motor Kanan
    apmKiri=map(apmKiri, APM_MIN_KIRI, APM_MAX_KIRI,
550,2400);
    apmKanan=map(apmKanan, APM_MIN_KANAN,
APM_MAX_KANAN, 550,2400);
    if (!ping_ok){
        apmKiri=  constrain(apmKiri,550,1400);
        apmKanan=  constrain(apmKanan,550,1400);
       }
    motorKiri.writeMicroseconds(apmKiri);//balik
    motorKanan.writeMicroseconds(apmKanan);//balik
  }
  if (mode==MODE_TRACKING){
         digitalWrite(11,HIGH);  //click
         digitalWrite(10,LOW);
         digitalWrite(9,HIGH);
       if ((Serial.available()>0)&&(ping_ok)){
          bacaRaspberry=Serial.read();
       }
          interruptAktif();
          kontrolTracking();
          kecepatanKanan ();
          kecepatanKiri ();
       if (!ping_ok){
          motorKanan.writeMicroseconds(1460);
          motorKiri.writeMicroseconds(1460);
       }
```

```
        else {
        motorKanan.writeMicroseconds(1500+potKanan);
           motorKiri.writeMicroseconds(1500+potKiri);
        }
           delay(10);
  }
  if (mode == MODE_RASPBERRY){
     interruptMati();
     apmKiri=avg(pulseIn (12, HIGH, 35000),3); //Pin 12, APM 1,
motor Kiri
     apmKanan=avg(pulseIn (13, HIGH, 35000),3); //Pin 13, APM 3,
motor Kanan
     motorKanan.writeMicroseconds(1460);//diam
     motorKiri.writeMicroseconds(1460);//diam
     potKiri=0; potKanan=0;
     if ((apmKiri < APM_MIN_KIRI +150)
         &&(apmKanan < APM_MIN_KANAN +150)
         &&(mode==MODE_RASPBERRY)){ //tuas mode X ke KIRI
raspberry
         digitalWrite(11,LOW);  //kiri
         digitalWrite(10,HIGH);
         digitalWrite(9,LOW);
      }
     else if ((apmKiri > APM_MAX_KIRI -500)
        &&(apmKanan > 1684 -100 )
        &&(apmKanan < 1684 +100 )
        &&(mode==MODE_RASPBERRY)){ //tuas mode X ke
KANAN raspberry
        digitalWrite(11,LOW);  //kanan
        digitalWrite(10,LOW);
        digitalWrite(9,HIGH);
        }
     else if ((apmKanan > APM_MAX_KANAN -500)
        &&(apmKiri > 1716 -100 )
        &&(apmKiri < 1716 +100 )
        &&(mode==MODE_RASPBERRY)){ //tuas mode Y ke ATAS
raspberry
         digitalWrite(11,LOW);  //atas
         digitalWrite(10,HIGH);
```

```
        digitalWrite(9,HIGH);
      }
    else if ((apmKiri > APM_MAX_KIRI -150)
       &&(apmKanan > APM_MAX_KANAN -150)
       &&(mode==MODE_RASPBERRY)){ //tuas mode Y ke
BAWAH raspberry
      digitalWrite(11,LOW);  //bawah
      digitalWrite(10,LOW);
      digitalWrite(9,LOW);
     }
    else{
      digitalWrite(11,HIGH);  //diam
      digitalWrite(10,LOW);
      digitalWrite(9,LOW);
    }
  }
}

void doEncoderKiri() {
  if ((digitalRead(encoderKiriPinA) == HIGH) && (digitalRead(4)==
HIGH)) {
    encoderKiriPos++;
  } else {
    encoderKiriPos--;
  }
}
void doEncoderKanan() {
  if ((digitalRead(encoderKananPinA) == HIGH) &&
(digitalRead(5)== HIGH)) {
    encoderKananPos++;
  } else {
    encoderKananPos--;
  }
}
void kecepatanKiri (){
  newpositionKiri = encoderKiriPos;
  newtimeKiri = micros();
  velKiri = ((oldpositionKiri-newpositionKiri)*1000000
/(newtimeKiri-oldtimeKiri));
```

```
  oldpositionKiri = newpositionKiri;
  oldtimeKiri = newtimeKiri;
  if (velKiri< velSetKiri){
    potKiri=potKiri+(0.1/*(velSetKiri-velKiri)*(100/maxSpeed)*/);
//kontrol Proporsional kecepatan
  }
  else if (velKiri> velSetKiri){
    potKiri=potKiri-(0.1/*(velKiri-velSetKiri)*(100/maxSpeed)*/);
//kontrol Proporsional kecepatan
  }
}
void kecepatanKanan (){
  newpositionKanan = encoderKananPos;
  newtimeKanan = micros();
  velKanan = ((newpositionKanan-oldpositionKanan)*1000000
/(newtimeKanan-oldtimeKanan));
  oldpositionKanan = newpositionKanan;
  oldtimeKanan = newtimeKanan;
  if (velKanan< velSetKanan){
    potKanan=potKanan+(0.1/*(velSetKanan-
velKanan)*(100/maxSpeed)*/);  //kontrol Proporsional kecepatan
  }
  else if (velKanan> velSetKanan){
    potKanan=potKanan-(0.1/*(velKanan-
velSetKanan)*(100/maxSpeed)*/);  //kontrol Proporsional kecepatan
  }
}
void kontrolTracking(){
  if (bacaRaspberry<=33){velSetKanan=0;velSetKiri=0;}
  else if ((bacaRaspberry<145)&&(bacaRaspberry>33)){//belok kiri
    //bacaRaspberry=145-bacaRaspberry;
    velSetKiri=(bacaRaspberry-34)*(maxSpeed/111);      //kontrol
Proporsional kecepatan
    velSetKanan=maxSpeed;
}
  else if (bacaRaspberry>145){//belok kanan
    //bacaRaspberry=255-bacaRaspberry;
    velSetKanan=(255-bacaRaspberry)*(maxSpeed/111);       //kontrol
Proporsional kecepatan
```

```
   velSetKiri=maxSpeed;
  }
}
//=analog input average==
double avg(double input, int count){
  double output=0;
  double avgbuff=0; //buffer untuk menyimpan data rata-rata
  for (int i=0; i<count; i++){
    avgbuff = input;
    output=output+avgbuff;
  }
  return output/(count);
}
//=Prosedur mengaktifkan interrupt
void interruptAktif(){
        if (interrupt_ok==0){
          attachInterrupt(0, doEncoderKiri, RISING);  // encoder pin on
interrupt 0 - pin 20
          attachInterrupt(1, doEncoderKanan, RISING);
          interrupt_ok=1;
        }
}
void interruptMati(){
        if (interrupt_ok==1){
          detachInterrupt(0);
          detachInterrupt(1);
          interrupt_ok=0;
        }
}
```

**Berikut ini program pada Raspberry Pi 2:**

```
/* Demo of modified Lucas-Kanade optical flow algorithm.
   See the printf below */

#ifdef _CH_
#pragma package <opencv>
#endif

#define CV_NO_BACKWARD_COMPATIBILITY
```

```
#ifndef _EiC
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <ctype.h>
#include <wiringPi.h>
#include <wiringSerial.h>
#endif

IplImage *image = 0, *grey = 0, *prev_grey = 0, *pyramid = 0,
*prev_pyramid = 0, *swap_temp;

int win_size = 10;
const int MAX_COUNT = 500;//?
CvPoint2D32f* points[2] = {0,0}, *swap_points;
char* status = 0;
int count = 0;
int need_to_init = 0;
int night_mode = 0;
int flags = 0;
int add_remove_pt = 0;
CvPoint pt;
CvPoint ptL=cvPoint(100,100);


void on_mouse( int event, int x, int y, int flags, void* param )
{
   if( !image )
      return;

   if( image->origin )
      y = image->height - y;

   if( event == CV_EVENT_LBUTTONDOWN )
   {
      pt = cvPoint(x,y);
      add_remove_pt = 1;
   }
```

```
}


int main( int argc, char** argv )
{
   wiringPiSetup();
   int fd = serialOpen("/dev/ttyACM0",9600);
   char getC;

   CvCapture* capture = 0;

   if( argc == 1 || (argc == 2 && strlen(argv[1]) == 1 &&
isdigit(argv[1][0])))
      capture = cvCaptureFromCAM( argc == 2 ? argv[1][0] - '0' : 0 );
   else if( argc == 2 )
      capture = cvCaptureFromAVI( argv[1] );

   if( !capture )
   {
      fprintf(stderr,"Could not initialize capturing...\n");
      return -1;
   }

   /* print a welcome message, and the OpenCV version */
   printf ("Welcome to lkdemo, using OpenCV version %s
(%d.%d.%d)\n",
      CV_VERSION,
      CV_MAJOR_VERSION, CV_MINOR_VERSION,
CV_SUBMINOR_VERSION);

   printf( "Hot keys: \n"
         "\tESC - quit the program\n"
         "\tr - auto-initialize tracking\n"
         "\tc - delete all the points\n"
         "\tn - switch the \"night\" mode on/off\n"
         "To add/remove a feature point click it\n" );

   cvNamedWindow( "LkDemo", 0 );
   cvSetMouseCallback( "LkDemo", on_mouse, 0 );
```

```
   for(;;)
   {
      IplImage* frame = 0;
      int i, k, c;

      frame = cvQueryFrame( capture );
      if( !frame )
         break;

      if( !image )
      {
         /* allocate all the buffers */
         image = cvCreateImage( cvGetSize(frame), 8, 3 );
         image->origin = frame->origin;
         grey = cvCreateImage( cvGetSize(frame), 8, 1 );
         prev_grey = cvCreateImage( cvGetSize(frame), 8, 1 );
         pyramid = cvCreateImage( cvGetSize(frame), 8, 1 );
         prev_pyramid = cvCreateImage( cvGetSize(frame), 8, 1 );
         points[0] =
(CvPoint2D32f*)cvAlloc(MAX_COUNT*sizeof(points[0][0]));//?
         points[1] =
(CvPoint2D32f*)cvAlloc(MAX_COUNT*sizeof(points[0][0]));//?
         status = (char*)cvAlloc(MAX_COUNT);//?
         flags = 0;//?
         ptL = cvPoint((frame->width)/2,(frame->height)/2);
      }

      cvCopy( frame, image, 0 );
      cvCvtColor( image, grey, CV_BGR2GRAY );

      if( night_mode )
         cvZero( image );

      if( need_to_init )
      {
         /* automatic initialization */
         IplImage* eig = cvCreateImage( cvGetSize(grey), 32, 1 );
         IplImage* temp = cvCreateImage( cvGetSize(grey), 32, 1 );
```

```
        double quality = 0.01;
        double min_distance = 10;

        count = MAX_COUNT;
        cvGoodFeaturesToTrack( grey, eig, temp, points[1], &count,
                            quality, min_distance, 0, 3, 0, 0.04 );
        cvFindCornerSubPix( grey, points[1], count,
           cvSize(win_size,win_size), cvSize(-1,-1),

cvTermCriteria(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS,20,0.0
3));
        cvReleaseImage( &eig );
        cvReleaseImage( &temp );

        add_remove_pt = 0;
     }
     else if( count > 0 )
     {
        cvCalcOpticalFlowPyrLK( prev_grey, grey, prev_pyramid,
pyramid,
           points[0], points[1], count, cvSize(win_size,win_size), 3,
status, 0,

cvTermCriteria(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS,20,0.0
3), flags );
        flags |= CV_LKFLOW_PYR_A_READY;
        for( i = k = 0; i < count; i++ )
        {
           if( add_remove_pt )
           {
              double dx = pt.x - points[1][i].x;
              double dy = pt.y - points[1][i].y;

              if( dx*dx + dy*dy <= 25 )
              {
                 add_remove_pt = 0;
                 continue;
              }
           }
```

```
            if( !status[i] )
               continue;

            points[1][k++] = points[1][i];
            cvCircle( image, cvPointFrom32f(points[1][i]), 3,
CV_RGB(0,255,0), -1, 8,0);
         }
         count = k;
      }

      cvCircle (image, ptL, 8, CV_RGB(255,0,0), 3, 8,0);
      cvRectangle(image,cvPoint(280,200),
               cvPoint(360,280),CV_RGB(255,255,255),1,8,0);
   if ((points[1][1].x>0)&&(points[1][1].x<200)) printf("z\n");
   else if(points[1][1].x>280) printf("x\n");
      if( add_remove_pt && count < MAX_COUNT )
      {
         points[1][count++] = cvPointTo32f(pt);
         cvFindCornerSubPix( grey, points[1] + count - 1, 1,
            cvSize(win_size,win_size), cvSize(-1,-1),

cvTermCriteria(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS,20,0.0
3));
         add_remove_pt = 0;
      }

      CV_SWAP( prev_grey, grey, swap_temp );
      CV_SWAP( prev_pyramid, pyramid, swap_temp );
      CV_SWAP( points[0], points[1], swap_points );
      need_to_init = 0;
      cvShowImage( "LkDemo", image );
      if (serialDataAvail(fd)>0) getC=serialGetchar(fd);
   switch( (char) getC )
      {
      case 'a':
         ptL.x=ptL.x-9;
         printf ("%c\n",getC);
         break;
```

```
    case 'w':
        ptL.y=ptL.y-9;
        break;
    case 's':
        ptL.y=ptL.y+9;
        break;
    case 'd':
        ptL.x=ptL.x+9;
        break;
    case 'q':
        pt = ptL;
        add_remove_pt = 1;
        break;
    case 'e':
        ptL=cvPoint((image->width)/2,(image->height)/2);
        break;
    default:
        break;
}
    c = cvWaitKey(10);
    if( (char)c == 27 )
        break;
    switch( (char) c )
    {
    case 'r':
        need_to_init = 1;
        break;
    case 'c':
        count = 0;
        break;
    case 'n':
        night_mode ^= 1;
        break;
    default:
        break;
    }
}

cvReleaseCapture( &capture );
```

```
    cvDestroyWindow("LkDemo");

    return 0;
}
#ifdef _EiC
main(1,"lkdemo.c");
#endif
```

**Berikut ini program pada Arduino Mega:**

```
#include <Servo.h>
#define thromin 1373//analog kanan bawah
#define thromax 2497//analog kanan atas
#define MODE_BIASA      1
#define MODE_RASPBERRY  2
#define MODE_TRACKING   3
int mode = MODE_BIASA;
int a;
int b = 90;
//char junk = ' ';
String inputString = "";         // a string to hold incoming data
boolean stringComplete = false;  // whether the string is complete

//<-------variable sensor Ping
unsigned long echo = 0;
int ultraSoundSignal = 38; // Ultrasound signal pin
unsigned long ultrasoundValue = 0;
int nilaiPING=0;
boolean ping_ok = 1;
boolean ambil_ok = 0;
boolean loop_ambil=0;

//<------variable motor servo
Servo  ser1,ser2,ser3,ser4,ser5,ser6,serCam;  //variabel  8  servo;  6
servo lengan 2 servo Tank

//<------variable receiver
int input1,input2,input3,input4,
input5,input6,input7,input8,input9; //input dari remote
int serIn;
```

```
int out1,out2,out3,out4,
out5,out6,out7,out8,out9; //output menuju servo
double serOut={14,90,3,90,90}; //arm initial position
int res1,res2,res3,res4,
res5,res6,res7,res8,res9;      //buffer      untuk      mengurangi
ketelitian/resolution
int res={0};

//<-------panjang lengan dalam cm
float a2=  10.3;
float a3=  9.7;
float a4=  15.7;
float d=   4.3;
//======================Theta 3=================
//input dalam radian,hasil dalam radian
double theta3 (double x0, double y0, double z0, double phi0){
//input dalam radian
 double r = (-1* (acos( ( ((sqrt((x0*x0)+(y0*y0)))-
a4*cos(phi0))*((sqrt((x0*x0)+(y0*y0)))- a4*cos(phi0)) + ((z0-d)-
a4*sin(phi0))*((z0-d)-a4*sin(phi0))  -  (a2*a2)-
(a3*a3))/(2*a2*a3))));
 if (isfinite(r)) return constrain(r,-2.44, 0);// pembatasan -140 s.d
0
}

//======================Theta 2===============
//input dalam radian,hasil dalam radian
double theta2 (double x0, double y0, double z0, double phi0){
 double  r  =  (atan2  (((z0-d)-
a4*sin(phi0)),((sqrt((x0*x0)+(y0*y0)))-  a4*cos(phi0)))-  atan2
((a3*sin(theta3(x0,y0,z0,phi0))),(a2+(a3*cos(theta3(x0,y0,z0,phi0
))))));
 if (isfinite(r)) return constrain(r,0, 1.57); // pembatasan 0 s.d 90
}

//======================Theta 4===============
//input dalam radian,hasil dalam radian
double theta4 (double x0, double y0, double z0, double phi0){
 double r = (phi0-theta2(x0,y0,z0,phi0) -theta3(x0,y0,z0,phi0));
```

```
  if (isfinite(r)) return constrain(r,-2.44, 0); //pembatasan -140 s.d
0
}
//======================Theta 1================
//input dalam radian,hasil dalam radian
double theta1(double x0, double y0){
  double r = (atan2 (y0,x0));
  if (isfinite(r)) return constrain(r,-1.57, 1.57); //pembatasan -180
s.d 180
}

void execute(double x0, double y0, double z0, double phi0){
     phi0=((phi0)*PI/180);
     ser1.write(170-(180/PI)*theta2(x0,y0,z0,phi0));
     ser2.write((180/PI)*theta2(x0,y0,z0,phi0));//tambah  180,  yg
bawah juga

     ser3.write(-(180/PI)*theta3(x0,y0,z0,phi0));//tambah 180
     ser4.write(-(180/PI)*theta4(x0,y0,z0,phi0));
  }
//=================PING SENSOR================
unsigned long ping()
{
  pinMode(ultraSoundSignal, OUTPUT); // Switch signalpin to
output
  digitalWrite(ultraSoundSignal, LOW); // Send low pulse
  delayMicroseconds(2); // Wait for 2 microseconds
  digitalWrite(ultraSoundSignal, HIGH); // Send high pulse
  delayMicroseconds(5); // Wait for 5 microseconds
  digitalWrite(ultraSoundSignal, LOW); // Holdoff
  pinMode(ultraSoundSignal, INPUT); // Switch signalpin to input
  digitalWrite(ultraSoundSignal, HIGH); // Turn on pullup resistor
  echo = pulseIn(ultraSoundSignal, HIGH); //Listen for echo
  ultrasoundValue = (echo / 58.138) *.39*2; //convert to CM then
to inches
  return ultrasoundValue;
}
//=================analog input average==============
double avg(double input, int count){
```

```
  double output=0;
  double avgbuff=0; //buffer untuk menyimpan data rata-rata
  for (int i=0; i<count; i++){
    avgbuff = input;
    output=output+avgbuff;
  }
  return output/(count);
}
//================SETUP=======================
void setup(){
  ser1.attach(8);//servo base // theta 2 reverse
  ser2.attach(9);//servo base // theta 2
  ser3.attach(10);// theta 3
  ser4.attach(11);// theta 4
  ser5.attach(12);// griper twist
  ser6.attach(13);// capit
  serCam.attach(39);//servo kamera
  Serial.begin(9600);
  inputString.reserve(200);
  //-------------------------pin switch kamera
  pinMode(46,OUTPUT);
  digitalWrite(46,HIGH);
  //-------------------------mode input pin 2-7
  for (int i=2; i<=7; i++){
    pinMode(i,INPUT);
    digitalWrite(i,HIGH);
  }
  //-------------------------initialisasi pin ultasonic, pin 38
  pinMode(ultraSoundSignal,OUTPUT);
  //-------------------------initial arm position
  execute(serOut[0],0,serOut[3],-90);
  //-------------------------pin Mode selection dan Ping Sensor
  pinMode(43, OUTPUT);      //UNO pin 8
  digitalWrite(43, LOW);
  pinMode(44, OUTPUT);      //UNO pin A4
  digitalWrite(44, LOW);
  pinMode(45, OUTPUT);      //UNO pin A5
  digitalWrite(45, LOW);
  //Serial.begin(9600);
```

```
}
//================LOOP=======================
void loop (){
    //--------Switch MODE
    serIn=avg(pulseIn (5, HIGH, 35000),3); //Pin 5, Receiver 6,
switch TRACKING MODE
    //Serial.println(serIn);
    if ((serIn < thromin +150 )&&(serIn>0)){  //saklar ke atas
      digitalWrite(46,HIGH);   //nyalakan kamera biasa
      mode = MODE_BIASA;       //mode biasa
        digitalWrite(44,LOW);
        digitalWrite(45,HIGH);
      }
    else if ((serIn > thromin +250 )&&(serIn < thromax -250 )) {
      digitalWrite(46,LOW);  //nyalakan kamera raspberry
      mode = MODE_RASPBERRY;  //ganti mode raspberry
        ser5.write(180);
        digitalWrite(44,HIGH);
        digitalWrite(45,LOW);
        ambil_ok=0;
        loop_ambil=0;
        serIn=avg(pulseIn (7, HIGH, 35000),3);
      }
    else if (serIn > thromax -150 ){
      digitalWrite(46,LOW);  //nyalakan kamera raspberry
      mode = MODE_TRACKING;
        digitalWrite(44,HIGH);
        digitalWrite(45,HIGH);
    }
    else {
      digitalWrite(46,HIGH);  //nyalakan kamera raspberry
      }

    //---------Manual navigation
    if (mode==MODE_BIASA){
          digitalWrite(43,HIGH);
       serIn[0]=avg(pulseIn (2, HIGH, 35000),3); //Pin 2, Receiver
3, Gerak X //============channel<<==1==2==5==>>dudah di
booking APM
```

```
      serIn[2]=avg(pulseIn (3, HIGH, 35000),3); //Pin 3, Receiver
4, Servo kamera
      serIn[3]=avg(pulseIn (4, HIGH, 35000),3); //Pin 4, Receiver
7, Gerak Z
      serIn=avg(pulseIn (6, HIGH, 35000),3); //Pin 6, Receiver 8,
Griper twist
      serIn=avg(pulseIn (7, HIGH, 35000),3); //Pin 7, Receiver 9,
Griper capit
       //--------Gerak X
      if (serIn[0] > thromax-300){
        serOut[0]=serOut[0]-0.5;
        serOut[0]=constrain(serOut[0],3, 22);
        execute(serOut[0],0,serOut[3],-90);//(x0,y0,z0,phi)//phi
pake -90
      }
      else if ((serIn[0] < thromin+300)&&(serIn[0]>0)){
        serOut[0]=serOut[0]+0.5;
        serOut[0]=constrain(serOut[0],3, 22);
        execute(serOut[0],0,serOut[3],-90);//(x0,y0,z0,phi)//phi
pake -90
      }
      else   execute(serOut[0],0,serOut[3],-90);//(x0,y0,z0,phi)//phi
pake -90

      //---------Gerak Z
      if (serIn[3] > thromax-300){
        serOut[3]=serOut[3]-1;
        serOut[3]=constrain(serOut[3],-20, 5);
        execute(serOut[0],0,serOut[3],-90);//(x0,y0,z0,phi)//phi
pake -90
      }
      else if ((serIn[3] < thromin+300)&&(serIn[3]>0)){
        serOut[3]=serOut[3]+0.5;
        serOut[3]=constrain(serOut[3],-20, 5);
        execute(serOut[0],0,serOut[3],-90);//(x0,y0,z0,phi)//phi
pake -90
        //if (serOut[3] < -7)digitalWrite(43,LOW);
        //else digitalWrite(43,HIGH);
      }
```

```
      else   execute(serOut[0],0,serOut[3],-90);//(x0,y0,z0,phi)//phi
pake -90

      //---------gripper twist
      if (serIn > thromax-400){
        serOut=serOut+5;
        serOut=constrain(serOut,0, 180);
        ser5.write(serOut);
      }
      else if ((serIn < thromin+400)&&(serIn>0)){
        serOut=serOut-5;
        serOut=constrain(serOut,0, 180);
        ser5.write(serOut);
      }

      //---------Capit
       if (serIn > thromax -150 ) ser6.write(180);
       else   if   ((serIn   <   thromin   +150   )&&(serIn>0))
ser6.write(110);
       else ser6.write(110);

     //--------gerakkan servo kamera
     if ((serIn[2]>0)&&(serIn[2]<thromin+200)){
        serOut[2]=serOut[2]+5;
        serOut[2]=constrain(serOut[2],0, 180);
        serCam.write(serOut[2]);
       }
     else if (serIn[2]>thromax-200){
        serOut[2]=serOut[2]-5;
        serOut[2]=constrain(serOut[2],0, 180);
        serCam.write(serOut[2]);
       }
   }
   if (mode==MODE_TRACKING){
     //---------PING, berhenti ketika jarak sudah
     nilaiPING = ping();
     if (nilaiPING < 12){
       digitalWrite(43,LOW);
       Serial.println("aa");//mengaktifkan fungsi estimasi orientasi
```

```
      //loop_ambil=1;
      if (stringComplete) {
        Serial.println(inputString);
        a= inputString.toInt();
        if (a>90) b=a-90; //kalkulasi sudut
        else b=a+90;
        ser5.write(b);//gerak gripper twist
        loop_ambil=1;
        // clear the string:
        inputString = "";
        stringComplete = false;
      }
      Serial.println("ok");
     //if (Serial.available()>0){
     }
     else digitalWrite(43,HIGH);
     //}
     while ((loop_ambil)&&(serIn>thromax-150)){
       if (!ambil_ok){
         serOut[3]=serOut[3]-0.05;
         if (serOut[3]<-18) {//-20
           ambil_ok=1;
           ser6.write(180);
         }
       }
       if (ambil_ok){
         serOut[3]=serOut[3]+0.05;
         if (serOut[3]>5) {//5
           serOut[3]=5;
           ser5.write(180);
           loop_ambil=0;
         }
       }
       execute(16,0,serOut[3],-90);
       serIn=avg(pulseIn (7, HIGH, 35000),3);
     }
   }
}
```

```
      void serialEvent() {
        while (Serial.available()) {
          // get the new byte:
          char inChar = (char)Serial.read();
          // add it to the inputString:
          inputString += inChar;
          // if the incoming character is a newline, set a flag
          // so the main loop can do something about it:
          if (inChar == '\n') {
            stringComplete = true;
          }
        }
}
```

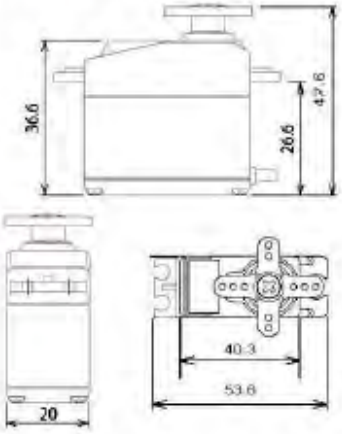
**Berikut ini program pada Raspberry Pi 3:**

```
import cv2
import numpy as np
import math
import serial
import time
import struct

angle = 0
angles = 0
a2=-1
cap = cv2.VideoCapture(0)
cap.set(3,320)
cap.set(4,240)
ser = serial.Serial(
 port = '/dev/ttyACM0',
 baudrate = 9600,
 parity = serial.PARITY_NONE,
 stopbits = serial.STOPBITS_ONE,
 bytesize = serial.EIGHTBITS,
 timeout = 1
)
while(1):
 x = str(ser.readline())
 print(x)
 if (x[2:4]=='aa'):
```

```
  ret,orimg = cap.read()
  orimg = cv2.flip(orimg,1)
  angle = 0
  hsv = cv2.cvtColor(orimg,cv2.COLOR_BGR2HSV)
  lower =
cv2.inRange(hsv,np.array([0,100,100]),np.array([10,255,255]))
  upper =
cv2.inRange(hsv,np.array([160,100,100]),np.array([179,255,255]))
  mask = cv2.addWeighted(lower,1,upper,1,3)
  kernel1 = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (4,4))
  kernel2 = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5,5))
  dst = cv2.morphologyEx(mask,cv2.MORPH_OPEN, kernel1)
  edge = cv2.blur(dst,(3,3))
  edge = cv2.Canny(edge,200,400)
  cdst = cv2.cvtColor(edge,cv2.COLOR_GRAY2BGR)
  lines = cv2.HoughLinesP(edge,1,math.pi/180,50,80,10)
  if(lines!=None):
   for x1,y1,x2,y2 in lines[0]:
    cv2.line(cdst,(x1,y1),(x2,y2),(0,255,0),3)
    angle = math.atan2(x1-x2,y1-y2)*(180/math.pi)
    angles = abs(angle)
    if(angles>90):
     angles = int(angles-90)
    else:
     angles = int(90+angles)
   cv2.imshow("line",cdst)
  print(angles)
  if (a2==angles):
   ser.write(str(angles).encode())
   time.sleep(1)
  ser.close()
  a2=angles
  ser.open()
 else:
  print('no serial')
 if cv2.waitKey(1) & 0xFF == ord('q'):
  break
  cv2.destroyAllWindows()
```

# MG996R High Torque
## Metal Gear Dual Ball Bearing Servo



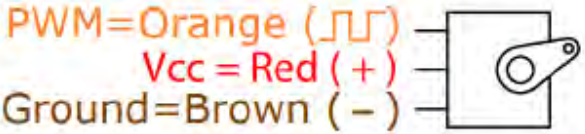

This High-Torque MG996R Digital Servo features metal gearing resulting in extra high 10kg stalling torque in a tiny package. The MG996R is essentially an upgraded version of the famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its predecessor. The gearing and motor have also been upgraded to improve dead bandwith and centering. The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

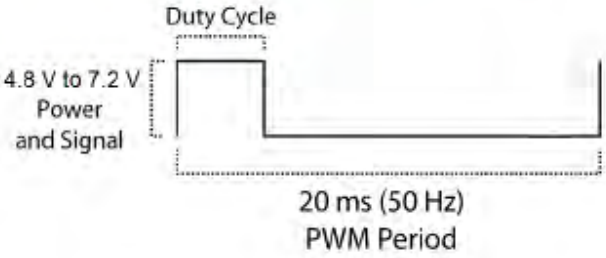
This high-torque standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG996R Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

**Specifications**

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kgf-cm (4.8 V ), 11 kgf-cm (6 V)
- Operating speed: 0.17 s/60° (4.8 V), 0.14 s/60° (6 V)

- Operating voltage: 4.8 V a 7.2 V
- Running Current 500 mA – 900 mA (6V)
- Stall Current 2.5 A (6V)
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC



*Halaman Ini Sengaja Dikosongkan*

# BAB V

# PENUTUP

## 5.1 Kesimpulan

Kesimpulan yang dapat diambil dari tugas akhir ini adalah:

1. Estimasi orientasi objek memiliki keasalahan terbesar sebesar -7,7%.
2. Batas nilai iluminansi pendeteksian orientasi objek adalah 7,1.
3. Kesalahan pergerakan griper dari orientasi objek yang terdeteksi adalah 0%.
4. Semakin jauh jarak objek dengan robot, semakin lama pula waktu pengambilan objek. Hal ini dikarenakan pergerakan robot dipengaruhi oleh waktu pendekatan robot menuju objek.
5. Dibutuhkan waktu pengambilan yang lebih lama pada orientasi objek 90 derajat dibandingkan orientasi objek yang lain. Hal ini dikarenakan pergerakan lengan jauh lebih lambat ketika orientasi bernilai 90 derajat.
6. Semakin besar jarak kamera sistem dengan objek, semakin besar kemungkinan kesalahan estimasi nilai orientasi. Hal ini sesuai dengan teori penggunaan fungsi *hough transform.*

## 5.2 Saran

Setelah pengujian dilakukan, terdapat beberapa saran untuk pengembangan sistem yang lebih baik, yaitu:

1. Penggunaan fungsi YCrCb dapat diterapkan untuk melihat seberapa presisi sistem jika dibandingkan penggunaan fungsi HSV.
2. Penggunaan kamera infrared dapat diterapkan agar sistem mampu mengenali objek pada waktu malam atau kondisi tanpa penerangan.
3. Penerapan *trackbar* guna pengaturan range filter warna dapat diterapkan pada penelitian selanjutnya.
4. Optimalisasi program untuk optimalisasi kuantitas unit proses.
5. Penggunaan unit proses yang memiliki kecepatan proses lebih baik, tentu mampu mengoptimalkan kinerja dari sistem. Sehingga, hal ini dapat diterapkan pada pengembangan penelitian selanjutnya.

*Halaman Ini Sengaja Dikosongkan*

# BIODATA PENULIS

Mohammad Nasrul Mubin lahir di Pekalongan, 18 September 1994. Anak kedua dari 3 bersaudara dari pasangan Furqon dan Nur Afiyah. Penulis menyelesaikan pendidikan dasar di SD Muhammadiyah Bligo 1, pendidikan menengah di SMPN 14 Pekalongan dan SMAN 1 Pekalongan. Tahun 2012, penulis memulai pendidikan sarjana di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Selama kuliah, penulis aktif dalam kegiatan organisasi, kepanitiaan, maupun asisten. Penulis aktif di BEM FTI ITS 2013/2014 sebagai staff dan menjabat sebagai Sekretaris Jenderal I pada periode 2014/2015. Kepanitiaan seperti ITS EXPO, FTI Olympic Games dan BARONAS pernah diikuti penulis. Penulis juga aktif sebagai asisten laboratorium Elektronika Dasar dan praktikum Elektronika.